

USER'S MANUAL

Machine Language Manager

for the
Bally® Arcade



The
Bit Fiddlers™

Computer Products
For Work and Play

P.O. Box 11023 • San Diego • California • 92111

(714) 565-1610

MACHINE LANGUAGE MANAGER

For The
BALLY® ARCADE

Copyright (C) 1982 by Andy Guevara

All Rights Reserved

Published by

THE BIT FIDDLERS™
P.O. Box 11023
San Diego, CA 92111-0010
(714) 565-1610

This program cartridge is meant to be used with the BALLY® ARCADE home video game. Any other use will void all warranties, expressed or implied.

REPLACEMENT--The MACHINE LANGUAGE MANAGER cartridge is warranted against all mechanical defects and workmanship for a period of one year. Should the cartridge cease to work during the warranty period, simply return the cartridge to The Bit Fiddlers for a free replacement. Direct all such returns and other correspondence to:

THE BIT FIDDLERS
3543 Armstrong St.
P.O.Box 11023
San Diego, CA 92111-0010

DISCLAIMER OF WARRANTIES AND LIABILITY

The Bit Fiddlers company makes no warranties, either expressed or implied, with respect to this manual or with the software described in this manual; its quality, performance, merchantability, or fitness for any particular purpose. The Bit Fiddlers company and program author shall have no liability or responsibility to purchaser or any other person or entity with respect to any liability, loss or damage caused or alleged to be caused by this software, including but not limited to any interruption of service, loss of business or anticipatory profits or consequential damages resulting from the use or operation of this software.

This manual is copyrighted. All rights reserved. This document may not, in whole or part, be copied, photocopied, reproduced, translated or reduced to any electronic medium or machine readable form without prior written consent from The Bit Fiddlers.

® BALLY is a registered trademark of AstroVision Inc.

TABLE OF CONTENTS

SUBJECT	PAGE
CHAPTER 1	
Introduction	1-1
CHAPTER 2	
Background Information	2-1
Hardware Overview	2-1
Z-80 Specifics	2-4
Conventions	2-5
Hexadecimal Notation	2-5
Assembler Notation	2-6
CHAPTER 3	
New Business	3-1
Startup Display	3-1
Number Keys	3-3
Command Keys	3-3
ADDR	3-3
WRITE	3-3
READ	3-4
LIST	3-4
INS	3-5
CALL	3-6
REG	3-7
*	3-8
*WRITE--Tape Output	3-8
*READ--Tape Input	3-8
*INS--Delete	3-9
*LIST--Print	3-9
*REG--Tape Display	3-10
CHAPTER 4	
Utility Programs	4-1
Screen Specification Program	4-1
Breakpoint Program	4-3
CHAPTER 5	
Using the Listing as a Source of Information	5-1
CHAPTER 6	
MLM Routines as Utilities	6-1
Clearing the Screen	6-1
Character Display	6-1
String Displays	6-2
Displaying the Values in a Register	6-3
Reading the Keypad	6-3
Changing the Screen Colors	6-4
Auto-Start Tapes	6-5

CHAPTER 7	
Sample Programs	7-1
"Critter" Program	7-2
Standard Color Generator	7-4
256 Color Display	7-5
ASCII Character Set	7-6
CHAPTER 8	
Quick Reference for MLM Commands	8-1
Command Sequences	8-1
Error Messages	8-2
CHAPTER 9	
Useful Memory Locations	9-1
MLM Locations	9-1
MLM Subroutines	9-1
Single Byte Calls	9-1
APPENDIX A: MLM SOURCE LISTING	A-1
APPENDIX B: Z-80 INSTRUCTION SET	B-1

CHAPTER ONE

INTRODUCTION

Congratulations! You have just stepped into the world of Machine Language. No longer are the secrets of fast graphics and infinite program control held out of your grasp.

But let's not be hasty! For along with this new found flexibility comes additional responsibilities and tedious frustrations you may not have experienced before. So, before we get too far along let me say a few words about who this new cartridge is aimed at.

You may have been aware of the efforts of many so-called "hackers" to break the bonds of BASIC and to add new and useful hardware and capabilities to the BALLY ARCADE. It is to these restless explorers that the Machine Language Manager is primarily dedicated.

But let me not dissuade the adventuresome who may not yet be familiar with the ways and wiles of machine language. There are a number of books available (some even written in English) that can take the uninitiated through the conventions used by nearly all microprocessors. A couple of good references are Programming the Z-80 by Rodney Zaks and Z-80 Software Gourmet Guide & Cookbook published by SCELBI. Personally, I use The Z-80 Handbook by ZILOG, mainly because they have an alphabetical and numeric listing of all the instructions and op-codes for the Z-80 in the back. For more specific information on the BALLY ARCADE, I strongly recommend the Bally On-Board ROM Sub-Routines manual available from the ARCADIAN newsletter. Of even more utility was the on-board ROM description by Dave Nutting Associates (143 pages worth, with the complete source listing thrown in on top), also available from the ARCADIAN. A lot of information used in creating the MLM was taken from the latter of these two publications.

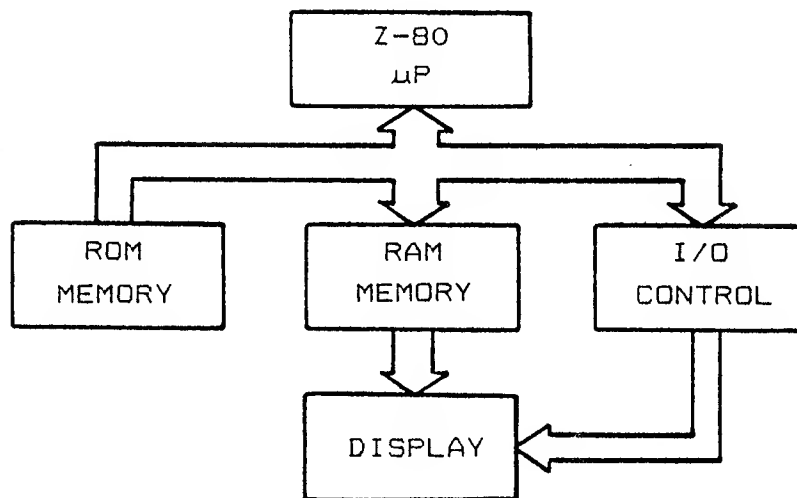
For those who wish to jump in without the aid of the aforementioned references, the next chapter holds a brief discussion of the conventions used throughout the rest of this manual.

CHAPTER TWO

BACKGROUND INFORMATION

Hardware Overview

First a few words about how the Bally Arcade is organized.



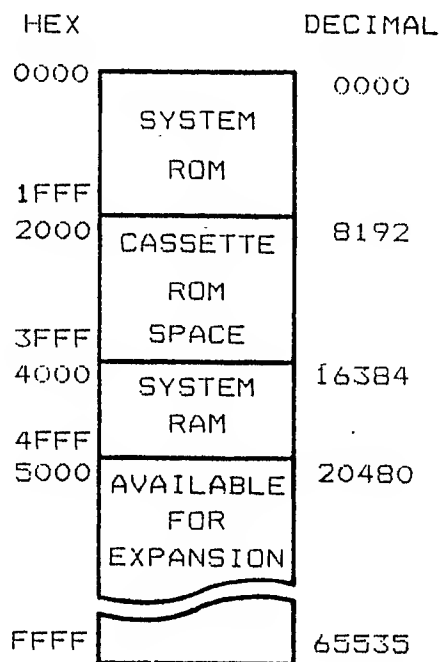
The above is a functional block diagram as I see it. The Z-80 microprocessor is directed by instructions (referred to as opcodes) taken from either Read-Only Memory (ROM) or Random Access Memory (RAM). Instructions and data in RAM are changeable, those in ROM are not. Why not all RAM, you ask? Data and instructions in ROM are permanent and don't go away when power is turned off. RAM, however, is "volatile" and goes blank when the power is removed.

The I/O control block represents all the circuitry needed for the Z-80 to make its actions known to the outside world and allows control actions in from outside. This includes reading and writing to the cassette tape port, screen image control, and reading the keypad and hand controls.

The ROM memory block represents all of the control memory that is dedicated to the system: the on-board games (like Gunfight), the animation and printout routines, and other housekeeping routines that are available to keep things flowing smoothly. Also included as part of the ROM block are the game cassettes which direct the workings of the on-board ROM and RAM. A small portion of the RAM is used by the on-board routines to keep track of variables, status, and the like.

All memory, both ROM and RAM, is made up of a continuous string of 8-bit clumps (bytes) any one of which can be made available to the Z-80. This is done by presenting memory with the unique 16 bit address of the desired byte. Don't worry about this point, the Z-80 does it automatically. All you have to do is get the right information in order. But more on this later.

There are over twenty thousand (4FFF Hexadecimal) bytes in the Bally Arcade arranged in the following manner.



The first 8K (thousand) are system ROM used as explained earlier. The next 8K are available for cassette ROM memory. Starting at byte 16384 (4000H) and continuing to 20479 (4FFFH) is system RAM memory. Beyond this is empty space available for those with the know-how to add more memory.

At this point the relationship between RAM and the TV screen should be discussed. Basically, the TV shares RAM with the system. That is, anything you wish to appear on the screen must be stored into RAM as a series of specific values. These values determine the colors of corresponding blocks (pixels) on the screen. Thus changing images is a matter of changing values in RAM. All of the RAM can be displayed on the screen.

But where do user programs go? Well, way back when the BALLY ARCADE was developed, memory was not as cheap as it is today, so they designed in the amount of memory they thought they could afford for the selling price of the ARCADE. In order to get reasonable graphics capability out of the limited amount of RAM, the developers opted to make all of the RAM displayable on the screen. This is great for ROM cassette programs, which need very little RAM, but not so good when you're trying to put programs into RAM. As a result, user programs must share the available RAM with the TV images. Bally Basic gets around this by using some tricks that result in half the memory being totally available to both screen and program, but limit displayable colors to two. The Machine Language Manager handles the RAM in a different way, separating it into two areas--program and graphics. This way we can have full use of all four colors for graphics. But more on this later.

Z-80 Specifics

To aid the Z-80 in executing a string of instructions in memory, a Program Counter (often "P-counter" or PC) holds the address that the Z-80 is working from and increments the address when the Z-80 needs the next instruction. Z-80 instructions can be 1, 2, 3, or 4 bytes long. The P-counter automatically keeps track of where the next legal instruction starts.

Speaking of legal instructions, there is no distinction between data bytes and instruction bytes in memory. This means that if you should, by some wild chance, direct the P-counter into the middle of a data table, the Z-80 won't know any better and will try to execute the data as if they were instructions. The results can be both beautiful and disastrous.

Besides the P-counter, there are other variable areas on the Z-80 chip known as "registers". These serve as intermediate storage and working areas. It is easier to shuffle data between them than between memory locations. They can hold 8 bits individually as the A, B, C, D, E, H, and L registers or operate on 16 bit quantities as the BC, DE, and HL Register Pairs. All 8 bit arithmetic and logical results are stored in the A register. Most 16 bit results are stored in H and L.

A	STATUS FLAGS
B	C
D	E
H	L
INDEX X	
INDEX Y	
PROGRAM COUNTER	
STACK POINTER	

Z-80 Register Set

For a deeper and better discussion of Z-80 registers and instruction set, see any of the previously listed references.

Conventions

The Machine Language Manager is a number-oriented tool, working directly with values the Z-80 can understand. So as to set things straight at the start, we will be using Hexadecimal (Base 16) numbers throughout this manual.

If you're not familiar with Hex representation, don't worry, it's not as bad as it sounds. In Decimal, each number position can hold a single digit value from 0 to 9. Hexadecimal extends this to 15 by using the letters A through F for the numbers beyond 9. Each number column then represents powers of 16, the same way Decimal columns represent powers of 10. Thus the number 12 in Decimal is 0C in Hex. Leading zeros are often used to avoid confusing Hex numbers with alphabetics. We'll also be putting an "H" on the end of Hex numbers to avoid confusion with Decimal numbers.

Converting numbers from Hex to Decimal is relatively easy. Take the number 123H. It can be represented as

$$1 \times 16^2 + 2 \times 16^1 + 3 \times 1$$

which is 291 Decimal.

There are charts available to help in conversion, if you really need to, in any self-respecting Machine Language book. The reason Hexadecimal is used is that numbers 0 through F can be represented by 4 bits with no leftovers.

For instance, the number 15 in Decimal can be shown in 4 bits as 1111. This Binary representation is what the Z-80 ultimately understands. Since the Z-80 uses 8 bit quantities, the whole thing would be 00001111. By splitting it into 4 bit segments again, we can show it as 0 for the first 4 bits, and F for the second using Hex notation. Thus the value of one byte can be shown as 2 Hex digits instead of 8 individual bits or a Decimal number between 0 and 255.

So much for math. Another convention worth noting is that of assembler code. An assembler is a program that uses quasi-English notation to put together a machine language program. Unfortunately, we are not privileged to have such a tool and can only work with numbers. We can still, however, put the notation to good use.

An example of assembler code:

LD A,B

Literally, "Load A from B", this means "copy the data out of register B into register A". Register B will still have its original value but A will also have B's value.

Most of the notation used is straight-forward (at least I think so) and will only be expanded upon if the function is not obvious from the notation.

A point of possible confusion should be noted here. When referring to an address, such as 4000H, the most significant byte will be shown first (40 00). When used in an instruction, such as a Jump to 4000H, Z-80 convention requires the least significant byte to be first:

OPCODE	INSTRUCTION	COMMENT
-----	-----	-----
C3 00 40	JP 4000H	;JUMP TO 4000H

This will always be true when representing any 2-byte quantity.

What you see under "OPCODE" are the Hex values for the instructions. This is the way we have to do things, since the Z-80 only understands numbers. In the example, the value "C3" was taken from one of the Z-80 books mentioned earlier. So, the "C3" is what the Z-80 knows as a "Jump" instruction. The two values after it are the address of where we want to jump, "lower" byte first.

I realize that this has been a pretty quick coverage of the Z-80 and probably doesn't answer many questions, but to adequately cover it would take more space than is available in a manual of this type. I strongly suggest that you go to an electronics store, a computer store, or just about any good bookstore and thumb through a few of the Z-80 books. When you find one that is readable and suits your needs, buy it and USE IT.

CHAPTER THREE

NEW BUSINESS

This is where we say Hello to those people who like to thumb through and skip over the introductory material. We'll try to explain the workings of the Machine Language Manager (MLM for short), in this chapter.

STARTUP

Put the cartridge in and press RESET. The screen should show:

```
0140 BYTES AVAILABLE STARTING AT 4E10
OK
```

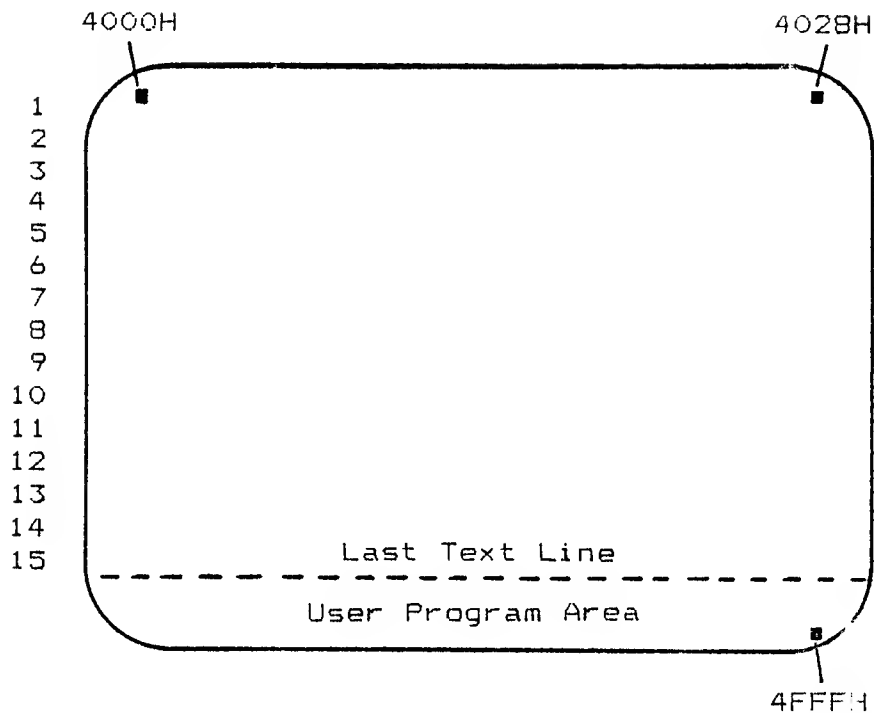
You've probably noticed that the characters look different and are a little smaller than Bally Basic's. In order to get more information on the screen and provide a reasonable LIST function, we opted to use our own character set rather than the one residing in the Bally system ROM. With this set we are able to put 39 characters on a single line.

Ok, so much for looks, what does it mean?

We've set up the screen RAM so that the display and program areas are physically separated. How much RAM is available for programming is up to the user (see Chapter 4 for specifics on changing it). Right now, the screen is set at 0FH (15 Decimal) text lines. This means that the screen will accomodate 15 lines of text before scrolling everything upward. Program memory starts at the first address available after the last text line of the screen. In this case it's address 4E10H. The amount of memory available is calculated from this address to the start of variables in RAM used by MLM and the Bally system. So this message says there are 140H "empty" bytes (320 Decimal) available for your use, starting at 4E10.

Decreasing the number of displayable text lines will increase the number of bytes available, and vice versa. For instance, setting the number of text lines at 3 will give 0C80H (3200 Decimal) free bytes available for your program. The impact of setting the screen text area to a certain length is to limit the space on the screen that will be cleared or scrolled, so that your program is not affected by these two operations. With text set at 3 lines, a Clear Screen operation will only wipe clean the area from the top of the screen to the bottom of the 3rd text line. The program area (starting where text line 4 usually is) will be left intact.

The minimum number of text lines allowed the user is 1. The maximum is 15.



Screen RAM starts at address 4000H and goes to 4FFFH. User area starts after last displayable text line.

While it is still possible to put programs into the text area, it's not advisable. Once the text starts scrolling, so will your program! The area set aside for programs is made immune to scrolling, screen clearing and accidental RESETs. Hit the RESET button a few times. The MLM knows if it's been awake so it doesn't try to set everything brand new. If you had a program in it, it would still be intact.

KEYS

Hopefully by now you will have put on the keypad overlay. Look at it closely. The numbers 0-9 and letters A-F are all Hexadecimal digits and have no other meanings. The keys on the right side and bottom are command keys.

NUMBERS

The MLM handles numbers in a different way than most systems. Each digit that is entered is shifted into the least significant digit of a 4 digit Input Register in memory. This register only remembers the last 4 digits entered and usually only needs to remember the last 2. In the absence of an ERASE or Clear Entry key this comes in handy. If an error is made, say you entered 25 when you wanted 2F, just enter what you meant to, in this case 2F, after the mistake and before pressing any of the command keys. For instance, enter the numbers:

12345AB

The numbers MLM will use for byte information are AB. The numbers used for address information are 45AB. The rest are thrown away.

The convention used by MLM commands that need number entries is to enter the numbers first, then the command key.

COMMANDS

ADDR

The ADDRESS key is used to tell the MLM where in memory you want the next operations to be performed. If you want to READ, WRITE, LIST or run a program at, say, 4E54H, you must begin with

4E54 **ADDR**

The ADDRESS key will put a colon (:) on the screen to show that this value has been loaded into the MLM Address Pointer.

WRITE

The WRITE key is used to put information into memory a byte at a time. For instance, let's say that at address 4E54H we wanted to put 3 bytes of data: 11, 22, 33. We would enter:

4E54 **ADDR** 11 **WRITE** 22 **WRITE** 33 **WRITE**

The screen would show:

4E54:11 22 33

MLM puts a space on the screen each time WRITE is pushed as a visual feedback that the command has been performed and to make the display more readable. The WRITE key also increments the Address Pointer automatically so that in the above example, 4E54H contains 11, 4E55 contains 22, and 4E56 contains 33 with the Address Pointer containing 4E57 for the next WRITE, READ or LIST.

READ

The READ key allows us to examine the contents of memory anywhere in the Bally system. Thus you can examine system ROM, or MLM (which is at address 2000H), or anywhere in RAM. For the example above, to verify that the data is really there, we would enter:

4E54 **ADDR** **READ** **READ** **READ**

The display will show:

4E54:11 22 33

Again, spaces are inserted and the Address Pointer is updated automatically.

LIST

The LIST key allows a more rapid and better formatted method of examining memory. In our example above, if no numbers have been entered since the last ADDR push, the numbers 4E54 should still be in the MLM Input Register, but the Address Pointer will have changed, pointing to successive bytes with each READ push. Pushing ADDR again should put 4E54 back into the Address Pointer. Type:

ADDR **LIST**

5000-4000 ADDRESS

The screen should show:

4E54:11 22 33 :
4E54: 11 22 33 00 "3."

The information following the contents of address 4E57 is the ASCII interpretation of the data in addresses 4E50 to 4E57. Codes that are not in the ASCII character set produce "." for this portion. The LIST command is organized to show 8 bytes at a time, breaking so as to start with xxx0: or xxx8:. Hit the LIST key again. It should show:

4E58: 00 00 00 00 00 00 00 00

The LIST command automatically updates the Address Pointer for the next 8 bytes.

Suppose you wanted to list a range of memory but didn't want to repeatedly push the LIST key. Type in the following:

2000 (ADDR) 2020 (LIST)

What you see is the first 40 bytes of MLM. Using this type of sequence you can list the entire contents of ROM and RAM with a single command! If perchance you should accidentally get caught in a longer list than you wanted or just want to stop a list in progress, hit the RESET button. That's right, the RESET button. Since RESET no longer clears memory, all it does is halt the operation in progress and returns control to MLM.

Do a list from 2628 to 26B8. This is where MLM messages are stored, and shows the use of the ASCII section of LIST.

(INS)

The "INS" key is for inserting a byte between two other bytes. This command and the DELETE command are the two most dangerous commands in MLM. To properly use the INSERT key, one step must be performed first: There is in MLM system RAM, a pointer called END. This pointer (address 4FC1) must have a value greater than the one in the Address Pointer to work properly. The purpose of END is to tell the INSERT and DELETE commands where the end of the program is. INSERT works by copying each byte in the range (from where the Address Pointer points to where END points), into the next higher location.

Type in the following:

4FC1 (ADDR) 57 (WRITE) 4E (WRITE)

4E54 (ADDR) 77 (INS)

4BC7

4C75

the first line points END at the proper byte, the second sticks a 77 at the beginning of our example above. Now LIST at 4E54:

4E54: 77 11 22 33 "3

END now has in it 4E58 (check it out). Whatever was in 4E57 is now gone. Note that END is updated automatically ONLY when INSERT and DELETE are used, so it should be kept track of and used with caution. Failing to do so can result in some strange things in your program.

Some tips:

If you're not strapped for memory space, it's good practice to sprinkle NOPs (NO-OP, opcode 00H), a do-nothing instruction, throughout your program. This is analogous to making line numbers in multiples of 10 in BASIC. It allows relatively painless expansion without having to move too many other instructions. The result is not having to use INSERT very often. Another good trick is to physically space subroutines and data tables away from the program. This allows the program room to grow.

```
4E50: -----  
      ----- Main Program  
      -----
```

```
4E60: RET
```

(EMPTY SPACE)

```
4E70: -----  
      ----- Subroutine  
      -----
```

```
4E7A: RET
```

```
4E80: -----  
      ----- Data table  
      -----
```

This way you can point END to one of the gaps, 4E61 in this case, and not have to change any of the call and table addresses in the main program as it expands.

One last bit of background about INSERT. The INSERT routine does a check to see if the Address Pointer is less than the contents of END. If END is smaller, it is changed to the value in the Address Pointer +1. This means that you can "build" your program using INSERT instead of WRITE and not have to worry about END, but only in this case. If you've just loaded in your routine from tape, END won't know where the end of your program is.

CALL

The CALL key is used to transfer control to a program or subroutine in memory. It's how you get your program to run. The convention for using it is to enter the address, then hit CALL. Simple, huh? For instance,

```
2347 CALL
```

calls MLM's clear screen routine at 2347.

The CALL routines in MLM also provide for the return process. That is, if the routine you've just transferred control to ends with an RET (Return, opcode C9), control will flow smoothly back to MLM when your routine is finished. If you don't want to give control back, well, that's your problem. To halt any program in progress, just hit RESET. If your program is scrambled at this point, it's in your code, not mine!

REG

This stands for "register", which is what this key acts on...the Z-80 register set. This command and associated subcommands allow you to preset values into the registers, as is usually needed during testing. For instance, a print subroutine may need the character value in the A register. By using this command, the A register may be preset to any value, after which the subroutine can be called to see the results. Pressing the REG key causes the keypad to alter its operations slightly. Hit the REG key. MLM will print:

A:

and wait. If you choose to change the present value of the A register, enter the value and press WRITE. If you choose not to, press READ. In either case MLM will print:

BC:

and you have the same options.

Some notes: READ does not produce the contents of A. To view the contents of the run-time registers, follow REG with LIST.

16 bit values like those required for BC are entered most significant byte first. That is, if BC is to have the value 4E54, then 4E54 is entered, followed by WRITE. This puts 4E in B and 54 in C.

The registers asked for, in order, are A, BC, DE, and HL. After HL is changed (but not after passing it using READ), MLM prints the present values of all the registers and present Address Pointer:

A:02 BC:0503 DE:4E50 HL:1004 ADDR:4E54

If no values were changed, pressing

REG LIST

would have given the same results.

To cancel the REG command at any point, simply press the LIST key. This is helpful if you only want to change the A register and skip the rest.

(*)

The last key on the overlay is marked "*". This is MLM's "shift" key, and it affects the other command keys in the following ways:

(*) **WRITE** --Tape Output

This combination opens the tape output port in much the same way as Bally Basic. Type in "*" and WRITE. MLM shows a red "T" to indicate Tape Output mode. Now do a list of a few lines. Notice how the print speed has slowed down? This is to accomodate the 300 Baud cassette rate. Note also that the ASCII portion is no longer shown. This is to keep MLM from reacting to false ADDR and WRITE commands. More on this later.

To record data onto tape, type in "*" and WRITE, then set up a list of the area you want saved. Before pressing LIST, start your recorder, allowing 1 or 2 seconds of "dead" time, then hit LIST. All the information sent to the screen is also sent to the tape.

To cancel the Tape Output mode once you've stored all your data, type in

(*) **WRITE**

again. MLM will respond with a green "T" signifying everything's back to normal.

(*) **READ** --Tape Input

Typing in this sequence opens the cassette tape input port. At this point you can read in an MLM-generated tape. MLM responds to colons (:) as Address commands, and spaces as Writes. This way, a listing on tape is all that's needed, and it maintains the listing format as it's read in.

MLM handles this function a little differently than Bally Basic. Basic's :INPUT makes the cursor go away, taking information only from the tape port. Pressing any key during :INPUT will cancel the tape input mode. MLM allows key inputs after *READ so that screen formatting, or whatever, can be done without having to re-open the tape input port. If you intend to use a full size ASCII keyboard, this allows commands to enter from either the keyboard or the keypad. However, only ADDRESS (colon) and WRITE (space) are supported. Note that this doesn't prevent you from writing your own keyboard driver program.

To cancel Tape Input mode, simply type "*". The green "*" says all is well again. RESET will also cancel this mode.

The Bit Fiddlers
P.O. Box 11023
San Diego, CA 92111-0010

April 6, 1982

Dear MLM Owner,

Since the introduction of the Machine Language Manager, we have been aware of the problem of cassette storage for those users who do not have access to the original Bally Basic 300 Baud cassette interface. It has been proposed that the 2000 Baud interface be incorporated into the MLM, and this remains the most logical answer to the problem.

There is, however, one minor problem with this solution. The ROMs used in producing the MLM have no more available programming space. This can be worked around using larger ROMs, but would require a whole new development cycle, for both hardware and software. This is not to say that we will leave MLM in its present state forever. Rather, it is to point out the fact that major changes in the product cannot appear before several months have elapsed.

In an effort to relieve this problem, we have developed a procedure for using MLM with Astrovision Basic's 2000 Baud cassette interface. This involves loading a tape using the Basic cartridge, then replacing it with the MLM cartridge. The full details are on the enclosed pages.

Admittedly, this is not the cleanest approach, but it has been found to be totally reliable. And much faster than 300 Baud.

An outgrowth of this method is that we are now able to produce tapes that auto-start when loaded using the 2000 Baud interface. Again, see the enclosed pages for details and limitations.

We hope that you find this information useful. We will continue to keep you informed of any future developments.

Sincerely,

A handwritten signature in cursive script, appearing to read "Andy Guevara".

Andy Guevara, Owner

MACHINE LANGUAGE MANAGER

UPDATE NUMBER 1

APRIL 6, 1982

PURPOSE:

The purpose of this update is to give users sufficient information to use the Machine Language Manager in cooperation with the Astrovision Basic 2000 Baud cassette interface. If you do not have the Astrovision Basic cartridge with built-in cassette interface, you do not need this update.

This entire update should be read carefully before experimenting with the programs provided. But don't worry, it's not as hard as it looks.

THEORY:

The theory behind the following procedure is that it is possible to put a program into a tight loop with interrupts locked out. This means that the system will not respond to keypad or joystick inputs, or even realize that the cartridge may have been removed from the unit. Therein lies the key.

The following short programs, based on this theory, allow you to maintain all the data in the Arcade while swapping one cartridge for another. Thus to store an MLM program using the 2000 Baud interface, you would run a specific short program, swap cartridges, start the recorder, then hit a specified key on the keypad. Once the key is hit, the program executes a jump into the area of the Astrovision Basic cartridge dealing with making recordings (the :PRINT program). In 25 seconds or so, the entire MLM environment, with the exception of register contents and color values, will have been recorded on tape.

The reverse situation, loading from tape, is similar in concept. There is a byte in memory where Basic expects to find an interrupt routine. It jumps to this routine when it has finished reading in a program tape, and before it tries to evaluate anything. If we replace this routine with information of our own, we can 1) put the machine into a tight loop so that we can swap cartridges, or 2) tell the machine that the interrupt routine is really the start of our main program. Since we have control of what information gets put on the tape, including how much memory the tape represents, we can do the above operations quite nicely.

STORING PROGRAMS ON TAPE:

A. THE PLAYBACK PROGRAM

When preparing to store your program on tape, you also have to consider how you're going to handle the playback process. Since all of screen memory gets put on tape, the playback program must be in memory when the tape gets made.

Entering the following program will take care of the playback process:

```
4E95:  AF          XOR A          ;CLEAR A TO 0
      D3 04      OUT (04),A    ;OUTPUT TO COLOR PORT 4
      C3 95 4E   JF 4E95H     ;JUMP BACK TO BEGINNING
```

Here's what's happening.

Astrovision Basic uses address 4E95H for storing its screen interrupt vector. Since interrupts are not responded to during the loading of a tape, the system will IMMEDIATELY respond once an Enable Interrupts instruction occurs. In Astrovision Basic, the EI happens just after the last byte has been read in.

So, once the last byte comes in, the system will execute an immediate jump to address 4E95H, starting the tight loop. The loop holds our data for us, so we can now take out the Astrovision Basic cartridge and put in MLM. That's all there is to it!

Line 2 is for correcting the background color. It can be extended to correct ports 6 and 7 for red and green, but this is left up to you.

No Disable Interrupts instruction is needed, because the DI is automatically issued by the system when answering an interrupt. So it's not necessary for us to do it.

B. THE RECORDING PROGRAM

Now that the proper playback program is in place, enter the following recording program:

```
4EA0:  F3          DI          ;DISABLE INTERRUPTS
      DB 17      IN A,(14H)  ;INPUT TO A FROM PORT 17H
      A7          AND A      ;TEST IF 0
      CA A0 4E   JF Z,4EA0H  ;IF SO GO BACK TO START
      3E C3      LD A,0C3H   ;OTHERWISE FIX MLM INTERRUPTS
      21 B1 4F   LD HL,4FB1H
      77          LD (HL),A
      22 B2 4F   LD (4FB2H),HL
      21 CA 0F   LD HL,0FCAH ;SET UP REGISTERS
      11 00 40   LD DE,4000H ;FOR :PRINT
      C3 48 20   JF 2048H    ;AND JUMP INTO :PRINT
```

The starting address of 4EA0H was chosen for convenience, being physically close to the playback code which MUST appear at 4E95H. Actually, the recording routine can be put anywhere in memory that is convenient for you. The only thing to be careful about is that the 'JF Z,' instruction in line 4, must always refer to the starting address of the routine. In this case it was 4EA0H.

What this program does is 1) Disable Interrupts, 2) checks to see if a key in the left column of the keypad is hit, and 3) if not, goes back to the Disable Interrupts instruction. If one of the left keys has been hit, it loads the registers with starting address and count information, then jumps into the :PRINT program in the Astrovision Basic cartridge. Lines 5 through 8 are a special case explained a little later.

For the :PRINT routine to work properly, it must have the DE and HL registers set with the right information. In order to save the entire MLM environment, DE must have the starting address of 4000H, and HL must have the number of bytes to save. In this case it is 0FCAH. 4FCAH is the last variable used by MLM, so that $4FCAH - 4000H = 0FCAH$ for the total number of bytes.

Lines 5, 6, 7, and 8 deal with keeping your program intact after the :PRINT program is done. What they do is write a small 'jump-self' program at MLM's interrupt vector location. Even though the cartridges will be changed, the system will retain the old vectors as long as the RESET button isn't hit. As a result of all this, once your recording is made, you can put MLM back in the Arcade, hit RESET, and be right back where you left off.

THE RECORDING PROCESS

When all of the above coding has been entered, you can follow this procedure:

1. Start the record program by typing its address, then CALL. In our case above, it would be 4EAO CALL.
2. Remove the MLM cartridge. Carefully.
3. Insert the Astrovision Basic cartridge, and connect the MIC cord to the cassette jack. DO NOT HIT RESET.
4. Start your recorder.
5. Hit the '*' key.

The recording process will be complete in less than 25 seconds. Timing it is the only way to know when it's done.

6. Put MLM back into the Arcade and hit RESET.

You should now be exactly where you were before you ran the program.

THE PLAYBACK PROCESS:

As you might have guessed, all we have to do to get our information back is to:

1. Put the Astrovision Basic cartridge in the system, and hit RESET. Connect the speaker output cord to the cassette jack in the cartridge.
2. Type in :INPUT or :RUN, either one. Don't type GO yet.
3. Cue the tape. Volume level should be about 9 on a scale of 10.
4. With the tape running, hit GO.
5. Once the tape is loaded, the LED will go dim or out, and the screen colors will change. DON'T HIT RESET YET.
6. Replace the Basic cartridge with MLM.
7. Now hit RESET. Your system should now be exactly as it was when you recorded it.

AUTO-START PROGRAM TAPES:

An auto-start program is one that starts as soon as the tape finishes loading. Automatically.

The obvious limitation on this is that the program cannot depend on MLM for any of its internal workings. That is, the program must be self-sufficient, and only use routines that are in the system ROM.

All that has to be done to make a program auto-start, once it has been adequately debugged, is put a jump instruction to the starting address of your program in address 4E95H in memory. For instance, if my program starts at 4E10H, at 4E95H I would type in:

```
4E95: C3 10 4E      JP 4E10H
```

From here I can make my recording as usual. But when the program is read back in, it will start automatically.

SPECIAL CONSIDERATIONS:

In setting up the system to record, any starting address and number of bytes can be used. The system will happily oblige.

This is particularly useful if you are using extended memory such as Viper or Blue Ram. The only thing to be careful about is that starting addresses other than 4000H must be specified before playing back the tape. This is easily accomplished by using the form :INPUT %(NNNNN), where NNNNN is the Decimal address of the program.

For instance, if our program starts at address 6000H, and we made the recording with this as the first address, then we would type in:

```
:INPUT %(24576)
```

24576 is the Decimal form of 6000H. If we were to simply type in :INPUT or :RUN, the program would be loaded at address 4000H.

One last limitation on this format. Due to the way that Basic stores information into screen RAM, this method will not work with addresses above 7FFFH (32767 Decimal). However, this doesn't prevent you from using 7FFFH as your starting point for both recording and playback.

SOME SUGGESTIONS:

We highly recommend trying out these procedures with no other programs in memory at the same time. At least none that aren't already saved somewhere else. Experimenting, when a program you've just spent 2 hours polishing is in memory, is inviting disaster.

Once you've mastered the process, it might be useful to keep a cassette with nothing else but these routines on it. That way, as a preparation for entering a new program, you could load in the tape routines first. That way they'll always be right, and ready for storing your interim results.

We also suggest using the routines as written, to save the entire MLM working environment each time. Since it only takes 25 seconds, setting up for a smaller area will be impractical in most cases.

Happy Recording.

(*) (INS) --Delete

If you haven't guessed by now, *INS produces the DELETE function. As discussed earlier in the INSERT section, it is very important to keep track of the END pointer. Aiming END at the last byte of your program +1 will take care of almost everything. I say "almost" because you still have to make sure that jumps and calls are made to the proper places.

Suppose that at 4E54 the following data was found:

4E54: 11 22 33 44 00

and you wish to remove the center 2 bytes.

Assuming that 44 is the last byte in your program:

4FC1 (ADDR) 58 (WRITE) 4E (WRITE) (puts 4E58 in END)

4E54 (ADDR) (READ) (*) (INS) (*) (INS) (deletes bytes 2 & 3)

The screen should now show:

4FC1:58 4E 4E54:11 *<*<

The "<" indicates something's been taken out.

Since no numbers have been entered, press ADDR, and READ or LIST the data area again. It should look like:

:
4E54: 11 44 00 00

END will have 4E56 in it.

An alternative to DELETE, if you're not too comfortable with it, is to change the errant values to 00 (NOP) using the WRITE command. Be sure, though, not to just zero out the first byte of a 2 or 3 byte instruction. They've all got to go.

(*) (LIST) --Print

For those of you who have a printer connected to the Bally, this action is the same as Bally Basic's *PRINT. Anything printed on the screen is sent to the printer port.

Actually, the printer and tape output ports are the same. The difference lies in the fact that *LIST filters out any unprintable characters before letting the data out the port. *WRITE has no filter and can send out any 8 bit quantity as a "character". However, most printers go bananas when you feed them the wrong codes. To cancel this mode, repeat *LIST.

(*) (REG) --Tape Display

The last one. This function fully duplicates the function of Bally Basic's :LIST. It is used to display information that's on tape without disturbing actual memory contents. This is helpful in locating one routine on a tape holding several, by watching for the addresses of the routine you're looking for as they're displayed. If you have a full ASCII keyboard, you could put the program title on tape before recording the program listing.

CHAPTER FOUR

UTILITY PROGRAMS

A couple of utility programs are included in MLM that do not operate as single keystrokes. For this reason they're discussed here separately.

Screen Specification Program

Called Screen Spec for short, it is most easily run by first hitting RESET, then CALL. What happens is that the address of Screen Spec is put into the MLM Input Register every time RESET is hit. Try it. The screen will print the following:

TEXT LINES:

Proper values are 1 through F. Zero acts the same as 1. Any number of numbers can be entered, only the last one will count. Entering nothing counts as entering F, which is the default value. End the input with WRITE.

The display should now show:

COLOR BOUNDARY:

The default value (if you don't enter anything) is 2CH. This value gets set into port 09, setting the right/left color boundary to the far right and giving us 4 colors to work with. Entering 12 will move the boundary to about the center of the screen and allow 4 colors for each side, for a total of 8. The upper 2 bits of the value entered into this port define which of the four colors for each side will frame the outside of the printable screen. Zeros indicate the frame will match the background color. 01 in the upper 2 bits will make the frame the same as the foreground color. Pressing WRITE ends this input.

The screen should now show:

BACKGROUND COLOR:

The default value is 00. This is the value for Black. If you don't like Black, key in any other number from 00 to FF. See if there's one you might like better. Blue is FA. End the input with WRITE.

MLM will now display:

FOREGROUND:

The choices are the same. Default in this case enters 07 into the color register for color #1. This is the value for White.

Once you've entered your choice and WRITE, MLM will clear the screen, limit the number of displayed text lines (and take care of scrolling properly), and set the foreground, background, and right/left parameters as you requested.

If you get into the middle of the program and decide that you didn't like an answer you already gave, or maybe don't want to change anything after all, just hit RESET. None of your choices will be entered into the system unless you go through the whole program.

To see the effect of this program, do the following:

- 1) Hit RESET, then CALL
- 2) After TEXT LINES: enter 5 then WRITE
- 3) Enter 3 more WRITES
- 4) Now put in the following program:

ADDR	OPCODE	INSTRUCTION	COMMENTS
-----	-----	-----	-----
4E50:	3E FF	LD A,OFFH	;LOAD A WITH OFFH
	D3 0A	OUT (0AH),A	;OUTPUT A TO PORT 0AH
	C9	RET	;GO BACK TO MLM

This will drop the "curtain" that hides the program area from being displayed. It's identical to $\$(10)=255$ in Bally Basic. Now run the program by typing

4E50 CALL

and you will see what was not affected by the Clear Screen operation. Hit list a couple of times to see the action of the scrolling function. To put it back to normal, hit RESET and 4 WRITES and everything will be as it was.

Breakpoint Program

Often (I know, I start a lot of things this way) when testing a program, the programmer finds that his routine causes the CPU to go off into Never-Never Land. To help him find out where the bad coding is, a breakpoint facility has been built into MLM.

The process for using breakpoints is simple: Find a place in your program that you wish to make sure that the CPU is getting to. Replace the opcode of that instruction with "CF". When (if?) the CPU runs across this code it will print the following:

```
BKPT ADDR: 1234          (Address where you put the breakpoint code)
A:11 BC:22 DE:33 HL:44 ADDR:5555
```

where the numbers indicate register values at the time the CPU ran across the breakpoint code. ADDR: indicates the contents of the Address Pointer.

At this point the register values should be checked against what you would expect at this point in your program. If all is well, replace the breakpoint code with your original opcode and put the breakpoint further downstream in your program.

CHAPTER FIVE

USING THE LISTING AS A SOURCE OF INFORMATION

There is a lot of good information and insight to be gained by just looking through the source listing of MLM. Granted, machine language is not exactly crystal clear to read and understand, even with a lot of comments by the author, but it can be valuable in a tight situation.

As an aid to understanding the interface between pure machine code and the Bally system ROM routines, a few good words are in order.

The Bally programmers, in their infinite wisdom, have made access to the bulk of the background tasks necessary in generating almost any game program relatively easy. These tasks include shape generation, animation, sound generation, timing and scoring among many others.

The method by which these on-board routines are called was made as universal as possible, so that different revisions of the system would be compatible. That is to say, normally a given subroutine is called by its address in memory, but when changes are made to a large program, the subroutine's address will also most likely change. The Bally people have found a way around this.

By way of the User Program Interface, any routine can call a system subroutine in the same way regardless of the revision level of the system.

To accomplish this feat, they have used the opcode "FF" as a sentinel to indicate a system call. This sentinel is immediately followed by the number of the subroutine to be called. For instance,

ADDR	OPCODE	INSTRUCTION	COMMENTS
----	-----	-----	-----
4E40:	01 03 05	LD BC,0503H	;RECTANGLE SIZE
	3E FF	LD A,0FFH	;COLOR MASK (GREEN)
	ED 5B C3 4F	LD DE,(COORD)	;WHERE
	FF	DEFB SYSTEM	;SYSTEM SENTINAL
	1C	DEFB RECTAN	;ROUTINE NUMBER
	C9	RET	

is one way to have the system draw a 3x5 pixel rectangle at the positions determined by the value in COORD (at 4FC3). The value in A is the color mask. That is, every byte used to draw the rectangle will have the binary value 11111111, meaning each pixel (whose color is defined by 2 bits) will be color #3, which was set to Green by MLM.

But back to the system interface. The alternative method of calling RECTAN is

ADDR	OPCODE	INSTRUCTION	COMMENTS
-----	-----	-----	-----
4E40:	FF	DEFB SYSSUK	; SENTINAL
	1D	DEFB RECTAN	; ROUTINE NO. PLUS 1
	10	DEFB 10H	; X COORD.
	10	DEFB 10H	; Y COORD.
	03	DEFB 03	; X SIZE
	05	DEFB 05	; Y SIZE
	FF	DEFB 0FFH	; COLOR MASK
	C9	RET	

Note that the sentinel is the same, but the routine number is different. Even-numbered routines expect the values to already be preset in the registers for use. Odd-numbered routines (1 greater than their counterparts) expect the variables to follow the subroutine number. This latter form is useful if none of the variables change, like walls or borders. If things need to be different the next time through, the first method is the way to go.

Note also the change in name of the sentinel from SYSTEM to SYSSUK. This was done to be consistent with the way Bally does things. SYSSUK uses an on-board subroutine called SUCK, which loads the Z-80 registers with the data following the sentinel. Which registers are used and what data is needed depends on the individual subroutine.

Subroutine 00 is of special interest. This is actually an additional sentinel that indicates entering an "interpreter" mode. Once entered, several system calls can be made successively without using the "FF" sentinel. Look at address 2026H in the listing. The subroutines FILL, SETOUT, MOVE BYTES, and COLSET are called in sequence, with no intermediate steps. The sentinals 02 and 03 (either one) are used to exit the interpreter and return to machine language.

A lot of processing can be done in just a few bytes by using system calls wherever possible.

CHAPTER SIX

MLM Routines As Utilities

In building MLM, it became apparent that a lot of the functions we were putting in would be very useful in user programs and would save a lot of programming time and space. This is why we include a full listing at the back of this manual, in case there's a routine you might need that we haven't explained.

Most, if not all, MLM routines can be coded as calls from within your programs or directly CALLED from the keypad. A more concise table of MLM routines can be found in Chapter 9.

Clearing The Screen

The subroutine for clearing the screen is at 2347H. It calculates how much to clear and only affects the text area, leaving the program area untouched. In your program:

```
      .      .  
      .      .  
      .      .  
CD 47 23  CALL CLEAR  
      .      .  
      .      .
```

From the keypad:

2347 CALL

Character Display

To display an ASCII character, the A register must be loaded with the ASCII value. Following this with opcode "D7" will display your character and return. For instance:

ADDR	OPCODE	INSTRUCTION	COMMENTS
----	-----	-----	-----
4E40:	3E 42	LD A,42H	;LOAD ASCII 'B'
	21 10 10	LD HL,1010H	;X,Y COORDS
	22 C3 4F	LD (COORD),HL	;STORE INTO MLM RAM
	D7	DEFB DISP	;CALL DISPLAY ROUTINE
	C9	RET	;GO BACK TO MLM

will place a "B" at coordinates (10,10) 16 pixels down and 16 over.

Once coordinates have been established, they are automatically updated for the next space to the right, and will do a carriage return and line feed when the edge of the screen is reached.

To print the larger 5x7 characters by way of the Bally system ROM, the following format must be used:

ADDR	OPCODE	INSTRUCTION	COMMENTS
----	-----	-----	-----
4E40:	11 10 10	LD DE,1010H	; X,Y COORDS
	0E 04	LD C,04H	; OPTIONS (0100)
	3E 42	LD A,'B'	; ASCII CHARACTER
	FF	DEFB SYSTEM	; SYSTEM SENTINAL
	32	DEFB CHRDIS	; DISPLAY ROUTINE
	C9	RET	

Or alternatively:

4E40:	FF	DEFB SYSSUK	; DATA IN LINE
	33	DEFB CHRDIS	; ROUTINE # PLUS 1
	10	DEFB 10H	; X COORD
	10	DEFB 10H	; Y COORD
	04	DEFB 0100B	; OPTIONS
	42	DEFB 'B'	; CHARACTER
	C9	RET	

The system ROM returns the updated coordinates in DE for you to change or keep as you see fit.

String Displays

To display a string of ASCII characters, like a word or a sentence, the following format should be used:

4E39:	DF	DEFB STRING	; CALL STRING
	40 4E	DEFW 4E40H	; MESSAGE ADDRESS
	C9	RET	
4E40:	48 45 4C 50 00		; ASCII 'HELP'

This will print "HELP" at wherever the Coordinate Pointer points. Coordinates are in 4FC3, X coordinate first. Each string must end with 00 to operate properly.

To print the same string in the larger character set, do the following:

```

4E50: 21 40 4E      LD HL,4E40H      ;MESSAGE ADDRESS
      11 10 10      LD DE,1010H      ;XY COORDS
      0E 04          LD C,04H        ;OPTIONS
      FF            DEFB SYSTEM
      34            DEFB STRDIS      ;STRING ROUTINE
      C9            RET

```

Or alternatively:

```

4E50: FF          DEFB SYSSUK      ;PARAMETERS TO FOLLOW
      35          DEFB STRDIS      ;ROUTINE #+1
      10 10        DEFW 1010H      ;XY COORDS
      04           DEFB 0100B      ;OPTIONS
      40 4E        DEFW 4E40H      ;MESSAGE ADDRESS
      C9           RET

```

In both cases, the string must end with 00.

Displaying the Value in a Register

Often the results of a calculation are to be displayed. This is helped along by use of the RGDIS routine at 245E.

RGDIS expects the value to be displayed to be in the A register, and the coordinates (at 4FC3) to be previously set. For instance, to show the contents of HL, the routine HLIST looks like this:

```

246F: 7C          LD A,H          ;H TO A
      CD 5E 24      CALL RGDIS      ;DISPLAY IT
      7D          LD A,L          ;L TO A
      CD 5E 24      CALL RGDIS      ;DISPLAY IT
      C9           RET            ;GO HOME

```

Reading the Keypad

To read the keypad, the routine KEYGET was written. What this routine does is scan the keypad until a key is pressed. It then evaluates the key to a number and returns with the key value in B. Note that nothing else can happen while KEYGET is running unless that something is interrupt driven.

Keys are organized as follows:

01	02	03	04
05	06	07	08
09	0A	0B	0C
0D	0E	0F	10
11	12	13	14
15	16	17	18

This routine prints out the key numbers:

```
4E40: CD 05 24      CALL CRLF          ;NEXT LINE
      CD 96 24      CALL KEYGET       ;ADDRESS 2496
      78            LD A,B            ;VALUE TO A
      CD 5E 24      CALL RGDIS        ;DISPLAY IT
      CD 51 24      CALL SPACE        ;PRINT A SPACE
      C3 43 4E      JP 4E43           ;LOOP BACK
```

To get out of it, hit RESET.

Changing the Screen Colors

There are a couple of ways to do this, depending on what you're trying to do. If you want to change the colors you're working with in MLM, say you wanted to change the Red and Green to Blue and Yellow, the easiest way is to change the values MLM holds in RAM, then call the routine that sets them into the I/O circuitry. The color list is at 4FC6, with the first color being color #3 (Green) and working backward to color #0 (Black). Load any values you like, then call subroutine M33 at address 25F2. This will take MLM through partial initialization, clearing the screen and putting up the opening message. Don't worry about your program getting messed up by this. The program area doesn't get touched by this operation.

That's one way. If you wanted to set the colors as part of your program, the following method works better. Somewhere in memory you'll have to put together a color list with color #3 being first. You can use MLM's area at 4FC6 if you are only going to use 4 colors.

From here you do a system call:

```
FF          DEFB SYSSUK
19          DEFB 19H          ;DO COLSET
C6 4F       DEFW COLIST      ;ADDRESS OF COLOR LIST
. .        (The rest of your program)
. .        .
```

To get a better idea, take a look at the Standard Color Generator program in Chapter 7.

A third method is by changing the color ports directly. There are 8 color ports--4 for each side of the screen. With the color boundary set to the far right side, only the last 4 will affect what you see on the screen. Ports 4, 5, 6, and 7 change colors 0, 1, 2, and 3 respectively, on the left side of the boundary. That is, any pixel of value 00 will have color 0, value 01 will have color 1, value 10 will have color 2, and value 11 will have color 3. At this point, a small program that loads the A register with the color wanted and outputs it to the proper port is all that's needed.

Auto-Start Tapes

This feature was added to MLM to allow self-starting programs to be loaded from tape. It's similar in action to having RUN be the last statement on a Bally Basic tape.

The steps necessary to make this happen are:

- 1) Record your program on tape in the manner outlined in Chapter 3.
- 2) After the last byte of information is on tape, with the tape still recording, type in

4FCA (ADDR)

then the starting address of your program, least significant byte first. That is, if your program starts at 4E20, then you would type in (colon meaning ADDR):

4FCA:20 4E

3) Type in the following (tape still recording):

```
4FAD:04
```

which sets the MLM operating mode to 4.

4) You can turn off your recorder now.

Now when you read in this tape, your program will start automatically.

Note:

It is important to turn off tape interrupts in the first part of your program so that they don't interfere with the way your program runs. The code for doing this is:

```
3E 08      LD A,08H      ;08 INTO A REGISTER
D3 0E      OUT (0EH),A    ;OUTPUT TO PORT 0E
```

The system will now ignore any interrupt requests from the tape port.

THE "CRITTER" PROGRAM

The following is a modified version of the "CRITTER" program of the Oct. 80 issue of CURSOR. Turn KN(1) to change the speed of the "critter".

```

4E40: F3      DI
      E5      PUSH HL
      21 4F 4E LD HL,4E4FH      ;LOAD SCREEN INTERRUPT VECTOR
      22 B2 4F LD (SCINT),HL   ;WITH ADDRESS OF ROUTINE
      CD 47 23 CALL CLEAR
      E1      POP HL
      FB      EI
      C9      RET

4E4F: F5      PUSH AF
      C5      PUSH BC
      D5      PUSH DE
      E5      PUSH HL
      DD E5   PUSH IX
      FD E5   PUSH IY
      DB 1C   IN A,(1CH)       ;GET KN(1) VALUE
      32 97 4E LD (4E97H),A    ;PUT IN VECTOR BLOCK (TIME BASE)
      FF      DEFB SYSTEM      ;START INTERPRETER
      00      DEFB INTPC
      07 75 4E MCALL (4E75H)    ;CALL VWRITE ROUTINE
      3F 95 4E 7C 4E VECT      ;MOVE VECTOR
      07 75 4E MCALL (4E75H)    ;CALL VWRITE
      02      DEFB EXIT        ;STOP INTERPRETER
      FD E1   POP IY           ;CLEAN UP AND GO HOME
      DD E1   POP IX
      E1      POP HL
      D1      POP DE
      C1      POP BC
      F1      POP AF
      FB      EI
4E73: C9      RET

4E75: 1F 95 4E 80 4E VWRITE
      0B

4E7C: 00 98      ;X BOUNDARIES
      00 50      ;Y BOUNDARIES
      00 00      ;PATTERN 0,0 POSITION
      02 0B      ;2 BYTE WIDE, 8 LINE LONG PATTERN SIZE
      0A A0 22 8B ;PATTERN
      AA AA 2A AB
      0B 20 20 0B
      0B 20 00 00

```

"Critic" program continued...

17 ;VECTOR BLOCK:

4E95:	20	;MAGIC REGISTER VALUE
	80	;VECTOR STATUS
	00	;TIME BASE
	05 00	;DELTA X
	00 00	;X POSITION
	03	;X CHECKS MASK
	05 00	;DELTA Y
	00 00	;Y POSITION
4EA1:	03	;Y CHECKS MASK

Standard Color Generator

This program generates 8 standard colors used in TV work.

```

4EAD: CD 47 23      CALL CLEAR      ; CLEAR SCREEN
      FF           DEFB SYSSUK      ; START INTERPRETER
      00           DEFB INTPC
      17           DO SETOUT
      B6           ; BLANK LINE
      13           ; R/L BOUNDS
      08           ; INTERRUPT LINE
      19           DO COLSET
      EB 4E        ; COLOR LIST AT 4EEB
      03           DEFB EXIT        ; STOP INTERPRETER
      01 6E 48     LD BC,486EH      ; WHITE BORDER
      3E 55        LD A,55H        ; COLOR #1
      11 14 06     LD DE, 0614H    ; POSITION
      FF           DEFB SYSTEM
      1C           DEFB RECTAN
      21 DA 4E     LD HL,4EDAH      ; TABLE ADDRESS
      16 09        LD D,09H        ; Y POSITION OF BARS
      06 08        LD B,08H        ; # OF BARS
4ECB: C5           PUSH B          ; SAVE IT
      7E           LD A,(HL)       ; GET COLOR FROM TABLE
      23           INC HL
      5E           LD E,(HL)       ; GET X POS'N
      23           INC HL
      01 0E 42     LD BC,420EH      ; BAR SIZE
      FF           DEFB SYSTEM
      1C           DEFB RECTAN
      C1           POP B           ; GET BACK COUNT
      10 F3        DJNZ 4ECBH      ; LOOP BACK
      C9           RET             ; ALL DONE

```

TABLE:

```

4EDA: 00 17      ; COLOR 00, AT X=17H
      55 24
      AA 33
      FF 3F
      00 4C
      55 57
      AA 64
      FF 71

```

COLOR LIST:

```

4EEB: AC          ; GREEN-----
      B6          ; YELLOW-----
      07          ; WHITE-----
      00          ; BLACK-----
      CD          ; CYAN-----
      5A          ; RED-----
      2B          ; MAGENTA-----
4EF2: F9          ; BLUE-----

      ]-- LEFT COLORS
      ]-- RIGHT COLORS

```


256 Color Program

This is a modified version of the program submitted by Jerry Burianyk in the Jan/Feb '81 issue of CURSOR. Turn KN(1) to change the number of displayed colors. To restore the screen, hit RESET, CALL, and four WRITES.

```

4EF5: F3      DI      ;
      F5      PUSH AF
      3E 4F   LD A,4FH ;LOAD SCREEN INTERRUPT VECTOR
      ED 47   LD I,A   ;WITH 4F10H
      3E 10   LD A,10H
      D3 0D   OUT (0DH),A
      CD 47 23 CALL CLEAR ;CLEAR SCREEN
      3E FF   LD A,0FFH ;SET INTERRUPT LINE
      D3 0F   OUT (0FH),A ;TO 256
      3E 12   LD A,12H  ;SET R/L BOUNDARY
      D3 09   OUT (09H),A ;TO MIDDLE OF SCREEN
      F1      POP AF
      FB      EI
      C9      RET

```

```

4F10: 12 4F      ;ADDRESS OF ROUTINE

```

```

4F12: F3      DI
      F5      PUSH AF
      D5      PUSH DE
4F15: DB 1C   IN A,(1CH) ;GET KN(1) VALUE
      D3 00   OUT (00),A ;SEND TO COLOR
      D3 01   OUT (01),A ;PORTS 0-3
      D3 02   OUT (02),A
      D3 03   OUT (03),A
      3D      DEC A      ;DECREMENT KN(1) VALUE
      20 F5   JR NZ,4F15H ;COUNT KN(1) VALUE TO ZERO
      D1      POP DE
      F1      POP AF
      FB      EI
4F25: C9      RET

```

ASCII Character Set

This little routine will print the entire MLM 3x5 pixel character set.

```
4F30: CD 05 24      CALL CRLF          ;GO TO NEXT LINE
      06 3B        LD B,3BH          ;SET UP COUNTER
      3E 20        LD A,20H          ;FIRST CHARACTER
HERE: D7           DEFB DISP          ;PRINT IT
      3C           INC A              ;GET NEXT ONE
      10 FC        DJNZ HERE          ;LOOP BACK
4F3B: C9           RET                ;GO HOME
```

A word of explanation...

DJNZ uses the B register as a counter. Whatever is in B, when this instruction takes place, is decremented. If the result is not zero, the displacement after the '10' opcode is added to the P-counter. In this case it's 0FCH or -4 Decimal. Since the P-counter has moved on to the next full instruction at 4F3B, the displacement is added to this number to produce the address 4F37 which we have dubbed "HERE".

If the result of decrementing the B register is zero, no displacement is added to the P-counter and execution continues with the next instruction.

CHAPTER EIGHT

QUICK REFERENCE FOR MLM COMMANDS

In the following examples, # stands for a number, 0 to F.

KEY means push indicated key.

ADDR

Sets Address Pointer to ####.

WRITE

Replaces byte at pointed address with ##, increments Address Pointer.

READ

Returns value of byte at pointed address, increments Address Pointer.

INS

Inserts ## at pointed address. Original contents of pointed address are moved to pointed address+1. Shifting continues until reaching address pointed to by End of File marker, END (address 4FC1H). END must be set prior to using INS.

ADDR **LIST**

Displays one line of data and ASCII interpretation of data starting at pointed address. Address Pointer is updated to next line.

LIST

Displays subsequent line. Address Pointer is updated.

####(1) **ADDR** ####(2) **LIST**

Displays data lines continuously starting at ####(1), continuing to ####(2).

REG

Makes run-time registers available for inspection and change.

WRITE changes indicated registers. **READ** skips to next register pair. **LIST** cancels REG command and lists register and Address Pointer contents.

CALL

Transfers CPU control to program at ####. If program ends with 'C9', control returns to MLM.

(*)

Alters command for certain keys.

(*) WRITE

Opens tape output port. All new data displayed on TV screen is output to port. Repeating (*) WRITE cancels this mode.

(*) READ

Opens tape input port. Programs created with *WRITE command will load properly. (*) (*) cancels this mode.

(*) INS

Deletes character at pointed address. END must be set prior to use. Data at pointed address+1 is shifted into pointed address. This continues until data pointed to by END is reached.

(*) LIST

Opens printer output port. All data displayed on TV screen is output to port. Retyping (*) LIST cancels this mode.

(*) REG

Opens tape input port for display. Data on tape is displayed without disturbing memory. (*) (*) cancels this mode.

RESET

Halts routine in progress. Resets interrupt vectors. Places address of Screen Specification program into Input Register for subsequent CALL command.

Error Messages

WARNING

Indicates next INS will force data into the MLM variable area.

ERR

Indicates the above situation has taken place.

CHAPTER NINE

USEFUL MEMORY LOCATIONS

LOCATIONS

4FAD	MODE BYTE
4FB0	LIGHT PEN INTERRUPT VECTOR
4FB2	SCREEN INTERRUPT VECTOR
4FB4	INPUT REGISTER
4FB6	ADDRESS POINTER
4FC1	END OF FILE MARKER
4FC3	Y-X COORDINATES FOR DISPLAY OUTPUT
4FC5	DISPLAY OPTIONS BYTE
4FC6	COLOR LIST
4FCA	ENTRY ADDRESS FOR MODE 4

SUBROUTINES

ADDRESS	NAME	DESCRIPTION
2347	CLEAR	CLEAR SCREEN ROUTINE
2405	CRLF	OUTPUT CARRIAGE RETURN, LINE FEED
23C3	GREEN	PRINT IN GREEN
23CB	RED	PRINT IN RED
23D0	NORM	PRINT IN NORMAL COLOR
2455	ERR	PRINT ERROR MESSAGE
2451	SPACE	PRINT A SPACE
246F	HLIST	DISPLAY CONTENTS OF HL
24C5	ININT	INITIALIZE INTERRUPTS
2496	KEYGET	GET KEYPAD ENTRY
245E	RGBDIS	DISPLAY CONTENTS OF A
23DE	SCROLL	SCROLL TEXT SCREEN

SINGLE BYTE CALLS

D7	DISP	DISPLAY CHARACTER IN A
DF	STRING	PRINT STRING LOCATED AT FOLLOWING ADDRESS
CF	BREAKPOINT	ENTER BREAKPOINT ROUTINE

APPENDIX A:

Machine Language Manager

Source Listing

ADDR	CODE	STMT	SOURCE	STATEMENT
0001			*****	
0002			*	
0003			BALLY ARCADE UTILITY PACKAGE	
0004			OR	
0005			"HOW TO GET STUFF IN AND OUT IN MACHINE LANGUAGE"	
0006			*	
0007			(C) 1981 ANDY GUEVARA	JULY 29, 1981
0008			*****	
0009			;	
0010			;	
0011			;	
>0013		0012	SKYD EQU	0013H
>00D7		0013	DISP EQU	0D7H ;RST 10
>00DF		0014	STRING EQU	0DFH ;RST 18
>00FF		0015	SYSSUK EQU	0FFH
>00FF		0016	SYSTEM EQU	0FFH
>4F50		0017	UPRAM EQU	4F50H ;BEGINNING OF U.P. RAM
		0018		;ROOM FOR STACK
		0019	;	
>4FAC		0020	ORG	4FACH
*4FAC 00		0021	UPSTK DEFB	0 ;U.P.STACK-GROWS DOWNWARD
*4FAD 00		0022	MODE DEFB	0 ;MODE REGISTER
*4FAE 00		0023	IOB DEFB	0 ;I/O INFORMATION BYTE
*4FAF 00		0024	KEYN DEFB	0 ;KEY INPUT NUMBER
*4FB0 0000		0025	LPINT DEFW	0 ;LIGHT PEN INTERRUPT VECTOR
*4FB2 0000		0026	SCINT DEFW	0 ;SCREEN INTERRUPT VECTOR
*4FB3 00		0027	IN1 DEFB	0 ;INPUT REGISTERS
*4FB5 00		0028	IN2 DEFB	0
*4FB6 00		0029	ADRG1 DEFB	0 ;ADDRESS REGISTERS
*4FB7 00		0030	ADRG2 DEFB	0
*4FB8 0000		0031	SCRN DEFW	0 ;SCREEN SIZE-BYTES
*4FBA 00		0032	SCRNLN DEFB	0 ;SCRATCH AT THIS LINE AND BELOW
*4FBB 00		0033	HRZCB DEFB	0 ;HORIZONTAL COLOR BOUNDARY
*4FBC 00		0034	LFLG DEFB	0 ;LIST FLAG
*4FBD 00		0035	RFG DEFB	0 ;REGISTER FLAG
*4FBE 0000		0036	PWRUP DEFW	0 ;POWER UP SIGNATURE
*4FC0 00		0037	PW2 DEFB	0 ;3RD BYTE
*4FC1 0000		0038	END DEFW	0 ;END OF FILE MARKER
*4FC3 0000		0039	COORD DEFW	0 ;XY COORDS FOR SCREEN
		0040		;D IS Y COORDINATE
*4FC5 00		0041	POPT DEFB	0 ;PRINT OPTIONS BYTE
*4FC6 00		0042	COLORS DEFB	0 ;COLOR 3
*4FC7 00		0043	C2 DEFB	0 ;COLOR 2
*4FC8 00		0044	FCOL DEFB	0 ;FOREGROUND COLOR (1)
*4FC9 00		0045	BCOL DEFB	0 ;BACKGROUND COLOR (0)
*4FCA 0000		0046	M4BA DEFW	0 ;ENTRY ADDRESS FOR MODE 4
>2000		0047	ORG	2000H
*2000 C31020'		0048	START JP	INIT
>2003		0049	DEFS	03 ;EMPTY SPACE
*2006 17		0050	DEFB	23 ;BUP REVISION LEVEL 2.3
*2007 C36E25'		0051	BPE JP	BRKPT ;BREAKPOINT ENTRY
*2008 C35923'		0052	DEP JP	DISPLAY ;DISPLAY ENTRY POINT
*200D C3B523'		0053	SEP JP	STRDIS ;STRING DISPLAY
*2010 F3		0054	INIT DI	;DISABLE INTERUPTS
*2011 31AC4F'		0055	LD	SP,UPSTK ;RESET STACK
*2014 2ABE4F'		0056	LD	HL,(PWRUP) ;CHECK IF BEEN AWAKE
*2017 1100AA		0057	LD	DE,0AA00H ;SIGNATURE VALUE
*201A B7		0058	OR	A ;CLEAR CARRY

ADDR	CODE	STMT	SOURCE	STATEMENT
*201B	ED52	0059	SBC	HL,DE
*201D	2007	0060	JR	NZ,STSPC ;BEEN ASLEEP, SET VALUES
		0061	;OK SD FAR	
*201F	3AC04F	0062	LD	A,(PW2) ;3RD BYTE
*2022	FEFA	0063	CP	0FAH
*2024	2828	0064	JR	Z,CONT ;JUMP IF OK
*2026	FF	0065	STSPC DEFB	SYSSUK ;ASLEEP, SET DEFAULT SPECS
*2027	00	0066	DEFB	00H ;START INTERPRETER
*2028	1B	0067	DEFB	1BH ;DO FILL
*2029	AD4F	0068	DEFW	MODE ;SYSTEM RAM START
*202B	2300	0069	DEFW	23H ;HOW MANY
*202D	00	0070	DEFB	00H ;WITH WHAT
*202E	17	0071	DEFB	17H ;DO SETOUT
*202F	B4	0072	DEFB	90.SHL.1 ;BLANK LINE 90 DECIM. AND BELOW
		0073		;THIS ALLOWS 360 BYTES OF RAM
*2030	2C	0074	DEFB	2CH ;44 DECIMAL--MOVE R/L BOUNDARY
		0075		;TO FAR RIGHT
*2031	08	0076	DEFB	8H ;INTERRUPT MODE 8
*2032	5F	0077	DEFB	5FH ;MOVE BYTES
*2033	C64F	0078	DEFW	COLORS ;TO WHERE
*2035	0400	0079	DEFW	04H ;HOW MANY
*2037	B926	0080	DEFW	COLIST ;FROM WHERE
*2039	19	0081	DEFB	19H ;COLSET--SET COLORS
*203A	C64F	0082	DEFW	COLORS ;ADDR OF COLOR LIST
*203C	03	0083	DEFB	03H ;EXIT INTERPRETER
*203D	21100E	0084	LD	HL,0E10H ;DEFAULT VALUE OF SCREEN SIZE
*2040	22B84F	0085	LD	(SCRN),HL
*2043	3E5A	0086	LD	A,90 ;90 DEC.
*2045	32BA4F	0087	LD	(SCRN),A ;VERT BLANK LINE
*2048	CD4723	0088	CALL	CLEAR ;WIPE SCREEN CLEAN
*204B	CD7824	0089	CALL	READY
*204E	FF	0090	CONT DEFB	SYSSUK
*204F	15	0091	DEFB	15H ;DO EMUSIC--STOP MUSIC
*2050	CDC524	0092	CALL	ININT ;SET UP INTERRUPTS
*2053	2100AA	0093	LD	HL,0AA00H ;WRITE POWER UP SIGNATURE
*2056	22BE4F	0094	LD	(PWRUP),HL
*2059	3EFA	0095	LD	A,0FAH
*205B	32C04F	0096	LD	(PW2),A
*205E	218225	0097	LD	HL,SCRSP ;SET UP CALL TO
*2061	22B44F	0098	LD	(IN1),HL ;SCREEN SPEC PROGRAM
*2064	CD1921	0099	CALL	INAD ;KEEPS LIST FROM RUNNING AWAY
*2067	AF	0100	XOR	A ;CLEAR FLAGS AND MODE
*2068	32AE4F	0101	LD	(IOB),A ;CLEAR I/O BYTE
*206B	32AD4F	0102	LD	(MODE),A
*206E	32C54F	0103	LD	(POPT),A
*2071	32BC4F	0104	LD	(LFLG),A
*2074	32BD4F	0105	LD	(RFG),A
*2077	CD0524	0106	CALL	CRLF
*207A	DF	0107	DEFB	STRING
*207B	B626	0108	DEFW	OKM ;SYSTEM PROMPT
		0109	;	
*207D	217D20	0110	MAIN LD	HL,MAIN ; MAIN LOOP
*2080	ES	0111	PUSH	HL ;THIS CAUSES RETURNS FROM
		0112		;MAIN ROUTINES TO RETURN HERE
*2081	CD9624	0113	CALL	KEYGET ;RETURNS WITH KEY NO. IN A
		0114		;USED AS A
		0115	;	;DISPLACEMENT IN JUMP TABLE
*2084	FF	0116	DEFB	SYSSUK ;INTERPRETER CALL

ADDR	CODE	STMT	SOURCE	STATEMENT
2005	5D	0117	DEFB	5DH ; INDEXB--BYTE TABLE LOOKUP
2006	9720	0118	DEFW	TTT-1 ; RETURNS WITH KEY TYPE IN
		0119		; HI NIBBLE
		0120		; TRANSLATION IN LOW NIBBLE
		0121		; TYPE=0 MEANS NUMBER,
		0122		; =1 MEANS COMMAND
		0123	; NOTE: THE -1 IS FOR THE ZERO ENTRY THAT DOESN'T EXIST	
2088	E60F	0124	AND	0FH ; ISOLATE TRANSLATION
208A	4F	0125	LD	C,A ; COPY DIGIT TO C
208B	AE	0126	XOR	(HL) ; PICK OUT TYPE
208C	CAF220	0127	JP	Z,NUMBER ; IF TYPE 0, SKIP AHEAD
208F	3AAD4F	0128	LD	A,(MODE) ; IF NOT THEN MUST BE TYPE 1
		0129		; FIND MODE
2092	FF	0130	DEFB	SYSSUK
2093	5B	0131	DEFB	5BH ; INDEXW--WORD TABLE LOOKUP
2094	B020	0132	DEFW	MDTBL ; MODE TABLE
2096	D5	0133	PUSH	DE
2097	C9	0134	RET	; SNEAKY WAY TO JUMP TO COMMAND
		0135		;
		0136		;
		0137	*****TYPE-TRANSLATE TABLE*****	
		0138		;
2098	0D	0139	TTT DEFB	0DH ; 1 D
2099	0E	0140	DEFB	0EH ; 2 E
209A	0F	0141	DEFB	0FH ; 3 F
209B	12	0142	DEFB	12H ; 4 CALL
209C	0A	0143	DEFB	0AH ; 5 A
209D	0B	0144	DEFB	0BH ; 6 B
209E	0C	0145	DEFB	0CH ; 7 C
209F	15	0146	DEFB	15H ; 8 REG (*TAPE DISPLAY)
20A0	07	0147	DEFB	07H ; 9 7
20A1	08	0148	DEFB	08H ; 10 8
20A2	09	0149	DEFB	09H ; 11 9
20A3	13	0150	DEFB	13H ; 12 LIST (*PRINT)
20A4	04	0151	DEFB	04H ; 13 4
20A5	05	0152	DEFB	05H ; 14 5
20A6	06	0153	DEFB	06H ; 15 6
20A7	17	0154	DEFB	17H ; 16 INS (*DELETE)
20A8	01	0155	DEFB	01H ; 17 1
20A9	02	0156	DEFB	02H ; 18 2
20AA	03	0157	DEFB	03H ; 19 3
20AB	11	0158	DEFB	11H ; 20 READ (*TAPE INPUT)
20AC	16	0159	DEFB	16H ; 21 *
20AD	00	0160	DEFB	00H ; 22 0
20AE	14	0161	DEFB	14H ; 23 WRITE (*TAPE OUTPUT)
20AF	10	0162	DEFB	10H ; 24 ADDR
		0163		;
		0164		;
		0165	*****MODE JUMP TABLE*****	
20B0	0021	0166	MDTBL DEFW	MODE0 ; NORMAL COMMANDS
20B2	0721	0167	DEFW	MODE1 ; REGISTER COMMANDS
20B4	BB22	0168	DEFW	MODE2 ; STAR (2ND SET) COMMANDS
20B6	9725	0169	DEFW	MODE3 ; SCREEN SPECIFICATIONS
20B8	4323	0170	DEFW	MODE4 ; EXTERNAL (USER) COMMANDS
		0171		;
		0172		;
		0173	*****JUMP TABLE FOR MODE 0*****	
		0174		;

ADDR	CODE	STMT	SOURCE	STATEMENT
		0175 ;		ROUTINE KEY VAL.
20BA	0E21	0176 MOJT	DEFW	ADDR ; 0, ADDRESS
20BC	2421	0177	DEFW	READ ; 1, READ AND INCREMENT
20BE	8F21	0178	DEFW	ACALL ; 2, CALL
20C0	AC21	0179	DEFW	LIST ; 3, LIST
20C2	3321	0180	DEFW	WRITE ; 4, WRITE TO MEMORY
20C4	0622	0181	DEFW	REG ; 5, LOAD REGISTERS
20C6	9D21	0182	DEFW	STAR0 ; 6, SET OPTIONS FLAG
20C8	4221	0183	DEFW	INS ; 7, INSERT IN MEMORY
		0184 ;		
		0185	*****MODE 1 JUMP TABLE*****	
		0186 ;		
		0187 ;		ROUTINE KEY VAL.
20CA	0D21	0188 M1JT	DEFW	M1RT ; 0, IGNORE 'ADDR' KEYPUSH
20CC	5122	0189	DEFW	RPLUS ; 1, SKIP TO NEXT REGISTER
20CE	0D21	0190	DEFW	M1RT ; 2, IGNORE 'CALL'
20D0	7322	0191	DEFW	RLIST ; 3, LIST REGISTERS
20D2	1B22	0192	DEFW	RWRT ; 4, CHANGE REGISTER CONTENTS
20D4	0D21	0193	DEFW	M1RT ; 5, IGNORE 2ND 'REG'
20D6	0D21	0194	DEFW	M1RT ; 6, IGNORE '*'
20D8	0D21	0195	DEFW	M1RT ; 7, IGNORE 'INS'
		0196 ;		
		0197	*****MODE 2 JUMP TABLE*****	
		0198 ;		
20DA	0E21	0199 M2JT	DEFW	ADDR ; 0
20DC	C222	0200	DEFW	TAPIN ; 1
20DE	8F21	0201	DEFW	ACALL ; 2
20E0	F822	0202	DEFW	PRINT ; 3
20E2	DD22	0203	DEFW	TAPOUT ; 4
20E4	D222	0204	DEFW	TADIS ; 5
20E6	1323	0205	DEFW	STAR2 ; 6
20E8	2523	0206	DEFW	DEL ; 7
		0207 ;		
		0208 ;		
		0209 ;		
		0210	*****MODE 3 JUMP TABLE*****	
20EA	A725	0211 M3JT	DEFW	M30
20EC	CA25	0212	DEFW	M31
20EE	DE25	0213	DEFW	M32
20F0	F225	0214	DEFW	M33
		0215 ;		
		0216 ;	NUMBER INPUT	
		0217 ;		
20F2	21B44F	0218	LD	HL, IN1 ; SET UP IN1
*20F5	79	0219	LD	A, C ; DIGIT IN LO HALF OF C
*20F6	ED6F	0220	RLD	
		0221	; RLD: A 4-BIT SHIFT AS FOLLOWS--	
		0222	; LOW A -> LOW IN1, LOW IN1 -> HI IN1, HI IN1 -> LOW A	
		0223	; THE EFFECT IS TO SHIFT INDIVIDUAL HEX DIGITS INTO IN1.	
		0224	; ONLY THE LAST 4 ARE REMEMBERED IN IN1 & IN2.	
		0225 ;		
*20F8	23	0226	INC	HL ; BINK POINTER (TO IN2)
*20F9	ED6F	0227	RLD	; DO IT AGAIN
*20FB	79	0228	LD	A, C ; RELOAD A WITH DIGIT
20FC	CDD423	0229	CALL	NUMDIS ; DISPLAY IT
*20FF	C9	0230	RET	; AND GO HOME
		0231 ;		
		0232 ;		

ADDR	CODE	STMT	SOURCE	STATEMENT
*2100	79	0233	MODE0	LD A,C ;MODE 0,GET DISPLACEMENT
*2	FF	0234		DEFB SYSSUK
*2102	5B	0235		DEFB 5BH ;INDEXW--WORD TABLE LOOKUP
2103	BA20	0236		DEFW MOJT ;JUMP TABLE ADDRESS
*2105	D5	0237		PUSH DE
*2106	C9	0238		RET ;JUMP TO COMMAND
		0239		;
*2107	79	0240	MODE1	LD A,C ;PUT DISPLACEMENT IN A
*2108	FF	0241		DEFB SYSSUK ;REG OPERATIONS
*2109	5B	0242		DEFB 5BH ;INDEXW
210A	CA20	0243		DEFW M1JT ;JUMP TABLE ADDRESS
*210C	D5	0244		PUSH DE ;JUMP TO PROCESS
*210D	C9	0245	M1RT	RET
		0246		;
		0247		;
		0248		;ADDRESS ROUTINE
		0249		;
210E	CD1921	0250	ADDR	CALL INAD ;MOVE DATA FROM INPUT TO ADDR REG
*2111	3E3A	0251		LD A,':' ;COLON TO A
*2113	D7	0252		DEFB DISP
*2114	AF	0253		XOR A ;CLEAR LIST FLAG
2115	32BC4F	0254		LD (LFLG),A
*2118	C9	0255		RET ;AND GO HOME
		0256		;
2119	11B64F	0257	INAD	LD DE,ADR61 ;MOVE ADDRESS TO ADR61
211C	21B44F	0258		LD HL,IN1 ;FROM INPUT REGISTER 1
*211	EDA0	0259		LDI ;SINGLE INSTRUCTION MOVE
*2121	EDA0	0260		LDI ;BETWEEN MEMORY LOCATIONS
		0261		;IT'S DONE TWICE BECAUSE TWO BYTES
		0262		;OF INFO ARE BEING MOVED
*2123	C9	0263		RET
		0264		;
2124	2AB64F	0265	READ	LD HL,(ADR61) ;LOAD ADDRESS
*2127	7E	0266		LD A,(HL) ;LOAD CONTENTS TO A
*2128	23	0267		INC HL ;INCREMENT CONTENTS
2129	22B64F	0268		LD (ADR61),HL ;STORE BACK
212C	CD5E24	0269		CALL RGDIS ;DISPLAY WHAT'S IN A
212F	CD5124	0270		CALL SPACE
*2132	C9	0271		RET
		0272		;
		0273		;WRITE TO MEMORY
		0274		;
2133	2AB64F	0275	WRITE	LD HL,(ADR61) ;PICK UP ADDRESS
2136	3AB44F	0276		LD A,(IN1) ;PICK UP INPUT BYTE
*2139	77	0277		LD (HL),A ;STORE IT
*213A	23	0278		INC HL ;BINK ADDRESS
213B	22B64F	0279		LD (ADR61),HL ;STORE IT BACK
213E	CD5124	0280		CALL SPACE ;OUTPUT A SPACE
*2141	C9	0281		RET ;GO HOME
		0282		;
		0283		;INSERT INTO MEMORY
		0284		;
		0285		;THIS ROUTINE MAKES USE OF AN END OF FILE MARKER (END)
		0286		;THAT SHOULD POINT TO THE LAST BYTE IN A PROGRAM +1.
		0287		;THE ROUTINE TESTS FOR ATTEMPTS TO WRITE INTO THE
		0288		;STACK, AND MAKES ALLOWANCES FOR ADDED MEMORY.
		0289		;IF 'END' IS LESS THAN THE PRESENT ADDRESS REGISTER
		0290		;CONTENTS, THE ROUTINE WILL UPDATE IT TO (ADR61)+1.

ADDR	CODE	STMT	SOURCE	STATEMENT
		0291 ;		
*2142	2AC14F'	0292 INS	LD	HL,(END) ;GET ADDRESS
*2145	11504F'	0293	LD	DE,UPRAM ;TEST IF PUSHING SYSTEM RAM
*2148	B7	0294	OR	A ;CLEAR CARRY FLAG FOR SBC
*2149	ED52	0295	SBC	HL,DE
*214B	381B	0296	JR	C,INS2 ;ADDRESS IS LESS THAN UPRAM
*214D	200B	0297	JR	NZ,INS1 ;ADDRESS IS GREATER
*214F	CDCB23'	0298	CALL	RED ;IF THE SAME ISSUE WARNING
*2152	DF	0299	DEFB	STRING
*2153	5926'	0300	DEFW	WAM
*2155	CDD023'	0301	CALL	NORM ;BACK TO NORMAL PRINT
*2158	180E	0302	JR	INS2 ;AND GO ON
*215A	11FF4F'	0303 INS1	LD	DE,4FFFH ;SEE IF INTO ADDED MEMORY
*215D	2AB64F'	0304	LD	HL,(ADRG1)
*2160	ED52	0305	SBC	HL,DE
*2162	3004	0306	JR	NC,INS2 ;OK IF POSITIVE
*2164	CD5524'	0307	CALL	ERR ;OTHERWISE OOPS
*2167	C9	0308	RET	
		0309 ;		
*2168	2AC14F'	0310 INS2	LD	HL,(END) ;GET THE END OF FILE
*216B	ED5BB64F'	0311	LD	DE,(ADRG1) ;SEE IF END>ADRG1
*216F	B7	0312	OR	A ;CLEAR CARRY
*2170	ED52	0313	SBC	HL,DE ;(HL)-(DE)
*2172	44	0314	LD	B,H
*2173	4D	0315	LD	C,L ;MOVE RESULT TO BC (COUNTER)
*2174	300B	0316	JR	NC,INS3 ;ALL SET
*2176	EB	0317	EX	DE,HL ;HL=(ADRG1)
*2177	23	0318	INC	HL ;OTHERWISE END=ADRG+1
*2178	22C14F'	0319	LD	(END),HL
*217B	010100	0320	LD	BC,01H ;SET COUNT TO 1
*217E	ED5BC14F'	0321 INS3	LD	DE,(END)
*2182	D5	0322	PUSH	DE
*2183	E1	0323	POP	HL
*2184	2B	0324	DEC	HL ;HL=END-1
*2185	EDB8	0325	LDDR	;COPY FROM (HL) TO (DE),
		0326		; (BC) TIMES
		0327		;AND DECREMENT EACH TIME.
*2187	CD3321'	0328	CALL	WRITE ;STUFF THE INFO
*218A	21C14F'	0329	LD	HL,END ;INC. END
*218D	34	0330	INC	(HL)
*218E	C9	0331	RET	;AND GO HOME
		0332 ;		
		0333 ;		
		0334 ;		EXECUTE A PROGRAM
		0335 ;		NOTE: CALL AND TAPE INPUT ARE INCOMPATIBLE FOR USE
		0336 ;		AT THE SAME TIME BECAUSE BOTH USE ALTERNATE REGISTER
		0337 ;		SET.
		0338 ;		
*218F	219A21'	0339 ACALL	LD	HL,CART ;SET UP RETURN PROCESS
*2192	E5	0340	PUSH	HL ;ON STACK
*2193	2AB44F'	0341	LD	HL,(IN1) ;GET ADDRESS OF PROGRAM
*2196	E5	0342	PUSH	HL
*2197	0B	0343	EX	AF,AF' ;EXCHANGE 'A' REGISTER
		0344		;FOR ITS ALTERNATE 'A' REGISTER
*2198	D9	0345	EXX	;EXCHANGE THE REST OF
		0346		;THE REGS FOR THEIR ALTERNATES
*2199	C9	0347	RET	;POP THE ADDRESS OFF THE STACK
		0348		;AND JUMP TO THE PROGRAM

ADDR	CODE	STMT	SOURCE	STATEMENT
		0349	;	
		0350	;	
*219A	08	0351	CART EX	AF,AF' ;CALL RETURN PROCESS
*219B	D9	0352	EXX	;GET BACK ORIG. SET
*219C	C9	0353	RET	;ALL DONE, GO HOME
		0354	;	
		0355	;	SET UP FOR MODE 2
		0356	;	
*219D	3E02	0357	STAR0 LD	A,02H
*219F	32AD4F'	0358	LD	(MODE),A ;SET MODE 2
*21A2	CDCB23'	0359	CALL	RED
21A5	3E2A	0360	LD	A,''
*21A7	D7	0361	DEFB	DISP ;TELL OPERATOR
*21A8	CDD023'	0362	CALL	NORM
*21AB	C9	0363	RET	
		0364	;	
		0365	;	
		0366	;	LIST ROUTINE
		0367	;	
		0368	;	LFLAG INDICATES BEING IN THE MIDDLE OF SUCCESSIVE SINGLE
		0369	;	LIST OPERATIONS. IT'S CLEARED BY THE ADDRESS KEY.
		0370	;	
*21AC	3ABC4F'	0371	LIST LD	A,(LFLG) ;TEST THE FLAG
*21AF	A7	0372	AND	A
*21B0	201D	0373	JR	NZ,OUTLN ;IF SET, DO A LINE AND GO HOME
		0374	;	
*21B1	3E0D	0375	LD	A,0DH ;OTHERWISE END THIS LINE
*21B4	D7	0376	DEFB	DISP
*21B5	ED5DB44F'	0377	LD	DE,(IN1) ;CHECK IF END ADDR WAS INPUT
*21B9	2AB64F'	0378	LD	HL,(ADRG1)
*21BC	ED52	0379	SBC	HL,DE
*21BE	280F	0380	JR	Z,OUTLN ;IF SAME, DO A LINE AND GO HOME
		0381	;	
		0382	;	MULTI-LINE LIST
		0383	;	
*21C0	CDCF21'	0384	MLIST CALL	OUTLN ;EA WAS INPUT, DO A LINE AND
		0385		;COME BACK.
		0386	;	
*21C3	2AB44F'	0387	LD	HL,(IN1) ;ENDING ADDRESS
*21C6	ED5BB64F'	0388	LD	DE,(ADRG1) ;NEW BEGINNING ADDRESS
*21CA	ED52	0389	SBC	HL,DE ;SUBTRACT BA FROM EA
*21CC	30F2	0390	JR	NC,MLIST ;IF POS, DO ANOTHER LINE
*21CE	C9	0391	RET	;ELSE, GO BACK
		0392	;	
		0393	;	OUTPUT A LINE
		0394	;	
*21CF	3EFF	0395	OUTLN LD	A,OFFH ;SET FLAG
*21D1	32BC4F'	0396	LD	(LFLG),A
*21D4	2AB64F'	0397	LD	HL,(ADRG1) ;PICK UP BEG. ADDR.
*21D7	CD6F24'	0398	CALL	HLIST ;DISPLAY 1ST BYTE ADDRESS
*21DA	3E3A	0399	LD	A,':'
*21DB	D7	0400	DEFB	DISP
*21DD	CD5124'	0401	CALL	SPACE
*21E0	CD2421'	0402	DL1 CALL	READ ;GET AND DISPLAY DATA
*21E3	7D	0403	LD	A,L ;CHECK FOR END OF LINE
*21E4	E607	0404	AND	07H ;BIT MASK
		0405		;IF 00 OR 0BH,
		0406		; THE LAST 3 BITS ARE ZERO

ADDR	CODE	STMT	SOURCE	STATEMENT
*21E6	20F8	0407	JR	NZ,OL1 ; IF NOT, GO BACK
		0408	;END OF LINE	
*21E8	3AAE4F	0409	LD	A,(IOB) ;CHECK IF TAPE OUT
*21EB	CB47	0410	BIT	0,A ;TURNED ON
*21ED	2802	0411	JR	Z,LASC ; IF NOT, DO ASCII PART
*21EF	1811	0412	JR	LA3 ;AND EXIT
		0413	;	
		0414	;LIST ASCII INTERPRETATION	
		0415	;	
*21F1	010800	0416	LASC	LD BC,08H ;BACK UP POINTER 8 BYTES
*21F4	ED42	0417	SBC	HL,BC
*21F6	41	0418	LD	B,C ;B=8 AS A COUNTER
*21F7	7E	0419	LA1	LD A,(HL) ;DISPLAY ASCII INTERPRETATION
*21F8	FE0D	0420	CP	ODH ;DON'T DISPLAY A CR
*21FA	2002	0421	JR	NZ,LA2
*21FC	3E2E	0422	LD	A,'.' ;DO A DOT INSTEAD
*21FE	D7	0423	LA2	DEFB DISP
*21FF	23	0424	INC	HL ;INC. POINTER
*2200	10F5	0425	DJNZ	LA1 ;DECR. COUNT & LOOP
*2202	3E0D	0426	LA3	LD A,ODH ;CARRIAGE RETURN
*2204	D7	0427	DEFB	DISP
*2205	C9	0428	RET	;GO HOME
		0429	;	
		0430	;LOAD REGISTERS	
		0431	;	
*2206	3E01	0432	REG	LD A,01H
*2208	32AD4F	0433	LD	(MODE),A ;SET MODE TO 1
*220B	DF	0434	DEFB	STRING
*220C	2826	0435	DEFW	AFM ;PUT OUT MESSAGE
*220E	AF	0436	XOR	A
*220F	32BD4F	0437	LD	(RFG),A ;TABLE INITIALIZATION
*2212	C9	0438	RET	
		0439	;	
		0440	;	
		0441	*****REGISTER TABLE*****	
		0442		ADDR RFG
*2213	2422	0443	RGTBL	DEFW AFG ; 0
*2215	2C22	0444		DEFW BCG ; 1
*2217	3822	0445		DEFW DEG ; 2
*2219	4422	0446		DEFW HLG ; 3
		0447	;	
		0448	;RFG OFFSET DETERMINES WHICH REGS TO PULL	
		0449	;	
		0450	;REGISTER WRITE ROUTINE	
		0451	;	
*221B	3ABD4F	0452	RWRT	LD A,(RFG)
*221E	FF	0453	DEFB	SYSSUK
*221F	5B	0454	DEFB	5BH ;INDEXW--WORD TABLE LOOKUP
*2220	1322	0455	DEFW	RGTBL ;GET JUMP ADDRESS
*2222	D5	0456	PUSH	DE ;SET UP JUMP
*2223	C9	0457	RET	;POP, JUMP
*2224	3AB44F	0458	AFG	LD A,(IN1) ;LOAD NEW VALUE
*2227	0B	0459	EX	AF,AF' ;PUT IN ALTERNATE A
*2228	CD5122	0460	CALL	RPLUS ;OUTPUT DESIGNATOR AND UPDATE RPL
*222B	C9	0461	RET	
*222C	ED4BB44F	0462	BCG	LD BC,(IN1) ;GET NEW VALUES
*2230	C5	0463	PUSH	BC
*2231	D9	0464	EXX	;EXCHANGE REGISTER SETS

ADDR	CODE	STMT	SOURCE	STATEMENT
2232	C1	0465	POP	BC ;PUT IN NEW VALUES
2233	D9	0466	EXX	;SWAP BACK
2234	CD5122'	0467	CALL	RPLUS
2237	C9	0468	RET	
2238	ED5BB44F'	0469	LD	DE, (IN1)
223C	D5	0470	PUSH	DE
223D	D9	0471	EXX	
223E	D1	0472	POP	DE
223F	D9	0473	EXX	
2240	CD5122'	0474	CALL	RPLUS
2243	C9	0475	RET	
2244	2AB44F'	0476	LD	HL, (IN1)
2247	E5	0477	PUSH	HL
2248	D9	0478	EXX	
2249	E1	0479	POP	HL
224A	D9	0480	EXX	
224B	AF	0481	XOR	A ;CLEAR A
224C	32BD4F'	0482	LD	(RFG), A ;CLEAR FLAG
224F	1822	0483	JR	RLIST ;LIST THE REG SET
		0484		;
2251	3ABD4F'	0485	RPLUS	LD A, (RFG)
2254	FE03	0486	CP	03H ;DONE YET?
2256	280C	0487	JR	Z, RP1 ;SKIP AHEAD IF SO
2258	3C	0488	INC	A
2259	32BD4F'	0489	LD	(RFG), A
225C	FF	0490	DEFB	SYSSUK
225D	5B	0491	DEFB	5BH ;INDEXW, GET MESSAGE ADDR
225E	6B22'	0492	DEFW	RMTBL ;REGISTER MSG TABLE
2260	CDBB23'	0493	CALL	STR1 ;PUT OUT MESSAGE
2263	C9	0494	RET	
2264	AF	0495	RP1	XOR A
2265	32BD4F'	0496	LD	(RFG), A ;CLEAR FLAG
2268	C3B022'	0497	JP	MOD0 ;GO HOME
		0498		;
		0499		;
226B	2826'	0500	RMTBL	DEFW AFM ;0
226D	2B26'	0501	DEFW	BCM ;1
226F	3026'	0502	DEFW	DEM ;2
2271	3526'	0503	DEFW	HLM ;3
		0504		;
		0505		;OUTPUT REGISTER CONTENTS
		0506		;
2273	AF	0507	RLIST	XOR A
2274	32BD4F'	0508	LD	(RFG), A ;CLR FLAG
2277	CD0524'	0509	CALL	CRLF
227A	DF	0510	DEFB	STRING
227B	2826'	0511	DEFW	AFM ;OUTPUT DESIGNATOR
227D	08	0512	EX	AF, AF' ;GET REG CONTENTS
227E	F5	0513	PUSH	AF
227F	08	0514	EX	AF, AF'
227C	F1	0515	POP	AF
227E	CD5E24'	0516	CALL	RGDIS
2284	CD5122'	0517	CALL	RPLUS ;DESIGNATOR & RFG UPDATE
2287	D9	0518	EXX	;ALTERNATE SET
2288	C5	0519	PUSH	BC ;SAVE REGISTER
2289	CDA422'	0520	CALL	SWDIS ;SWAP AND DISPLAY
228C	D5	0521	PUSH	DE
228D	CDA422'	0522	CALL	SWDIS

ADDR	CODE	STMT	SOURCE	STATEMENT
2290	E5	0523	PUSH	HL
2291	CDA422	0524	CALL	SWDIS
2294	D9	0525	EXX	; BACK TO NORMAL SET
2295	DF	0526	DEFB	STRING
2296	3A26	0527	DEFW	ADM
2298	2AB64F	0528	LD	HL, (ADRG1)
229B	CD6F24	0529	CALL	HLIST ; DISPLAY ADDRESS REGISTER
229E	CD0524	0530	CALL	CRLF
22A1	C3B022	0531	JP	MODE
		0532		;
		0533		; SWAP AND DISPLAY A REGISTER
		0534		;
22A4	D9	0535	SWDIS	EXX ; NORMAL SET
22A5	D1	0536	POP	DE
22A6	E1	0537	POP	HL ; GET CONTENTS
22A7	D5	0538	PUSH	DE ; PUT BACK RETURN ADDR
22A8	CD6F24	0539	CALL	HLIST ; SHOW CONTENTS
22AB	CD5122	0540	CALL	RPLUS ; DESIG. AND RFG UPDATE
22AE	D9	0541	EXX	; ALTERNATE SET
22AF	C9	0542	RET	
		0543		;
		0544		; CLEAN UP AND RESET MODE 0
22B0	AF	0545	MODE	XOR A ; CLEAR A TO ZERO
22B1	32AD4F	0546	LD	(MODE), A ; SET MODE TO 0
22B4	32BC4F	0547	LD	(LFLG), A ; CLEAR LIST FLAG
22B7	32C54F	0548	LD	(POPT), A ; CLEAR PRINT OPTONS
22BA	C9	0549	RET	; GO HOME
		0550		;
		0551		*****MODE 2 OPERATIONS*****
		0552		;
>22BB		0553	MODE2	
22BB	79	0554	LD	A, C ; GET DISPLACEMENT
22BC	FF	0555	DEFB	SYSSUK
22BD	5B	0556	DEFB	5BH ; INDEXW
22BE	DA20	0557	DEFW	M2JT ; JUMP TABLE
22C0	D5	0558	PUSH	DE
22C1	C9	0559	RET	; POP, JUMP
		0560		;
		0561		;
>22C2		0562	TAPIN	
22C2	F3	0563	DI	; INITIALIZE INTERRUPTS
22C3	D9	0564	EXX	
22C4	06FC	0565	LD	B, 0FCH ; SET COUNT
22C6	D9	0566	EXX	
22C7	3E18	0567	LD	A, 18H ; INT. ENABLE AND MODE
22C9	D30E	0568	OUT	(0EH), A
22CB	FB	0569	EI	
22CC	CD0524	0570	CALL	CRLF
22CF	C3B022	0571	JP	MODE ; CLEAN UP AND GO HOME
		0572		;
>22D2		0573	TADIS	
22D2	CDC222	0574	CALL	TAPIN ; INIT INTERRUPTS
22D5	21AE4F	0575	LD	HL, IOB ; SET TAPE DISPLAY BIT
22D8	CBDE	0576	SET	3, (HL)
22DA	C3B022	0577	JP	MODE
		0578		;
		0579		;
		0580		; WRITE TO TAPE

ADDR	CODE	STMT	SOURCE	STATEMENT
		0581	;	
>D		0582	TAPOUT	
22DD	21AE4F	0583	LD	HL,IOB ;GET I/O INFO
22E0	CB46	0584	BIT	0,(HL) ;TAPE OUT SET?
22E2	2009	0585	JR	NZ,TO1 ;YES, CLEAR IT
22E4	CB06	0586	SET	0,(HL) ;NO, SET IT
22E6	CB8E	0587	RES	1,(HL) ;CLEAR PRINT BIT
22E8	CDCB23	0588	CALL	RED
22EB	1805	0589	JR	TO2
22ED	CB86	0590	RES	0,(HL)
22EF	CDC323	0591	CALL	GREEN
22F2	3E54	0592	LD	A,'T'
22F4	D7	0593	DEFB	DISP
22F5	C3B022	0594	JP	MOD0
		0595	;	
		0596	;	
>22F8		0597	PRINT	
22F8	21AE4F	0598	LD	HL,IOB ;GET I/O BYTE
22FB	CB4E	0599	BIT	1,(HL) ;PRINT BIT CLEAR?
22FD	2807	0600	JR	Z,PR1 ;YES, SET IT
22FF	CB8E	0601	RES:	1,(HL) ;NO, CLEAR IT
2301	CDC323	0602	CALL	GREEN
2304	1807	0603	JR	PR2
2306	CDCB23	0604	CALL	RED
2309	CBCE	0605	SET	1,(HL) ;SET PRINT BIT
230B	CB86	0606	RES	0,(HL) ;CLEAR TAPE BIT
2	3E50	0607	LD	A,'P'
230F	D7	0608	DEFB	DISP
2310	C3B022	0609	JP	MOD0 ;GO HOME
		0610	;	
		0611	;	
>2313		0612	STAR2	
2313	3E08	0613	LD	A,08H ;DISABLE TAPE INPUT
2315	D30E	0614	OUT	(0EH),A
2317	21AE4F	0615	LD	HL,IOB ;GET I/O INFO
231A	CB9E	0616	RES	3,(HL) ;CLEAR TAPE DISPLAY MODE
231C	CDC323	0617	CALL	GREEN
231F	3E2A	0618	LD	A,'*'
2321	D7	0619	DEFB	DISP
2322	C3B022	0620	JP	MOD0
		0621	;	
		0622	;	
		0623	DELETE	
		0624	;	
>2325		0625	DEL	
2325	B7	0626	OR	A ;FIX CARRY FLAG
2326	2AC14F	0627	LD	HL,(END)
2329	ED5BB64F	0628	LD	DE,(ADRG1)
232D	ED52	0629	SBC	HL,DE ;GET NUMBER TO MOVE
232F	44	0630	LD	B,H
2330	4D	0631	LD	C,L ;COUNT TO BC
2	62	0632	LD	H,D
2332	6B	0633	LD	L,E
2333	23	0634	INC	HL ;(ADRG1)+1 TO HL
2334	EDB0	0635	LDIR	;COPY FROM (HL) TO (DE),
		0636		; (BC) TIMES
2336	3E3C	0637	LD	A,'<' ;A MARKER
2338	CDCB23	0638	CALL	RED

ADDR	CODE	STMT	SOURCE	STATEMENT
233B	D7	0639	DEFB	DISP
233C	21C14F	0640	LD	HL,END ;UPDATE END
233F	35	0641	DEC	(HL)
2340	C3B022	0642	JP	MOD0 ;GO HOME
		0643	;*****MODE 4 OPERATIONS*****	
		0644	;	
		0645	;MODE 4 IS AVAILABLE FOR REDEFINING THE KEYPAD FOR	
		0646	;OTHER USES. USER MUST ENTER THE ADDRESS OF HIS	
		0647	;KEYPAD HANDLING ROUTINE IN M4BA (ADDRESS 4FC8),	
		0648	;LOWER HALF FIRST. SETTING THE MODE TO 4	
		0649	;IN THE STARTUP ROUTINE WILL ROUTE CONTROL IN THIS	
		0650	;DIRECTION.	
		0651	;	
>2343		0652	MODE4	
2343	2ACA4F	0653	LD	HL,(M4BA) ;GET ENTRY ADDRESS
2346	E9	0654	JP	(HL) ;JUMP TO IT
		0655	;	
		0656	;	
		0657	;*****DISPLAY CONTROL ROUTINES*****	
		0658	;	
>2347		0659	CLEAR	
2347	ED4BB84F	0660	LD	BC,(SCRN) ;SIZE OF SCREEN
234B	110040	0661	LD	DE,4000H ;START OF SCREEN
234E	3E00	0662	LD	A,0H ;DATA TO FILL WITH
2350	FF	0663	DEFB	SYSTEM
2351	1A	0664	DEFB	1AH ;DO NT FILL--CLEAR SCREEN
2352	210000	0665	LD	HL,00H
2355	22C34F	0666	LD	(COORD),HL ;SET COORDINATES TO 0,0
2358	C9	0667	RET	
		0668	;	
		0669	;DISPLAY IS THE GENERAL DISPLAY ROUTINE FOR THE	
		0670	;SYSTEM. IT TAKES CARE OF THE SCREEN, TAPE, AND PRINTER	
		0671	;PORTS. PRINT DIFFERS FROM TAPE ONLY IN THAT	
		0672	;NONPRINTABLES ARE FILTERED OUT AND LINE FEEDS	
		0673	;ARE INSERTED.	
		0674	;	
>2359		0675	DISPLAY	
2359	E5	0676	PUSH	HL
235A	D5	0677	PUSH	DE
235B	C5	0678	PUSH	BC
235C	F5	0679	PUSH	AF
235D	21AE4F	0680	LD	HL,IOB ;TEST FOR TAPE OR PRINT OUT
2360	CB46	0681	BIT	0,(HL) ;TAPE OUT BIT
2362	C42724	0682	CALL	NZ,TWRT ;DO IT
		0683	;SIMPLE DISPLAY	
2365	F1	0684	POP	AF ;RESTORE CHARACTER
2366	F5	0685	PUSH	AF ;PUT IT BACK
2367	FE0D	0686	CP	ODH ;CHECK IF CARR. RETURN
2369	200A	0687	JR	NZ,D1 ;IF NOT, GO AHEAD
236B	CB4E	0688	BIT	1,(HL) ;PRINT ON?
236D	C42724	0689	CALL	NZ,TWRT ;YES, SEND CR TO PRINTER
2370	CD0524	0690	CALL	CRLF ;DO IT ON SCREEN
2373	182F	0691	JR	DRET
2375	FE20	0692	CP	20H ;CHECK IF PRINTABLE
2377	3804	0693	JR	C,DOT ;JUMP IF MINUS
2379	FE5B	0694	CP	5BH ;UPPER LIMIT
237B	3802	0695	JR	C,OK ;JUMP IF NEG.
237D	3E2E	0696	LD	A,'.' ;DO A DOT IF NOT PRINTABLE

ADDR	CODE	STMT	SOURCE	STATEMENT
237F	F5	0697	OK	PUSH AF ;SAVE IT
2	CB4E	0698		BIT 1,(HL) ;PRINT ON?
2382	C42724	0699		CALL NZ,TWRT ;YEP, DO IT
2385	ED5BC34F	0700		LD DE,(COORD) ;GET X,Y COORDINATES
2389	3AC54F	0701		LD A,(POPT) ;PICK UP OPTIONS, IF ANY
238C	4F	0702		LD C,A ;PUT IN C
238D	A7	0703		AND A
238E	2002	0704		JR NZ,OK1 ;IF NONZERO, USE IT
2390	0E04	0705		LD C,04H ;OTHERWISE STANDARD PRINT
2392	F1	0706	OK1	POP AF ;GET CHAR BACK
2393	EE80	0707		XOR 80H ;ALTERNATE FONT INDICATOR
2395	DD21BD26	0708		LD IX,SMLFNT ;SMALL CHAR. FONT DESCRIPTOR
2399	FF	0709		DEFB SYSTEM
239A	32	0710		DEFB 32H ;CHRDIS--OUTPUT CHARACTER ROUTINE
		0711		;RETURNS WITH UPDATED XY COORDS
		0712		;IN DE, D=Y, E=X
239B	3E9B	0713		LD A,9BH ;TEST FOR END OF LINE
239D	BB	0714		CP E
239E	DC1024	0715		CALL C,CLF ;IF SO,SET UP NEXT LINE
23A1	CDA923	0716		CALL CON ;PAINT CURSOR
23A4	F1	0717	DRET	POP AF ;AND GO HOME
23A5	C1	0718		POP BC
23A6	D1	0719		POP DE
23A7	E1	0720		POP HL
23AB	C9	0721		RET
		0722		;
2	010305	0723	CON	LD BC,0503H ;PAINT 3X5 CURSOR
23AC	3E55	0724		LD A,55H ;BINARY 01010101 COLOR MASK
		0725		; (COLOR #1)
23AE	ED53C34F	0726		LD (COORD),DE ;STORE COORDS BACK
23B2	FF	0727		DEFB SYSTEM
23B3	1C	0728		DEFB 1CH ;RECTAN--PAINT RECTANGLE
23B4	C9	0729		RET
		0730		;
		0731		;NOTE: STRING CLOBBERS ALMOST EVERYTHING, USE WITH CARE
		0732		;
>23B5		0733	STRDIS	
23B5	E3	0734		EX (SP),HL ;GET ADDRESS FROM CALLING ROUTINE
23B6	5E	0735		LD E,(HL)
23B7	23	0736		INC HL
23B8	56	0737		LD D,(HL)
23B9	23	0738		INC HL
23BA	E3	0739		EX (SP),HL ;FIX THE STACK
23BB	1A	0740	STR1	LD A,(DE) ;PICK UP CHARACTER
23BC	FE00	0741		CP 00H ;IF NULL, END
23BE	C8	0742		RET Z
23BF	D7	0743		DEFB DISP ;DISPLAY CHARACTER
23C0	13	0744		INC DE ;INCREMENT ADDRESS
23C1	1BF8	0745		JR STR1 ;DO ANOTHER
		0746		;
		0747		;COLOR CHANGES
		0748		;
23C3	F5	0749	GREEN	PUSH AF
23C4	3E0C	0750		LD A,0CH ;GREEN ON BLACK OPTION
23C6	32C54F	0751	G1	LD (POPT),A
23C9	F1	0752		POP AF
23CA	C9	0753		RET
		0754		;

ADDR	CODE	STMT	SOURCE	STATEMENT
23CB	F5	0755	RED	PUSH AF
23CC	3E08	0756		LD A,08H ;RED ON BLACK
23CE	18F6	0757		JR G1
		0758		;
23D0	F5	0759	NORM	PUSH AF
23D1	AF	0760		XOR A ;BACK TO NORMAL
23D2	18F2	0761		JR G1
		0762		;
		0763		; NUMERICAL DISPLAY ROUTINE
		0764		;
23D4	FE0A	0765	NUMDIS	CP 0AH ;CHECK IF WITHIN NUMERICAL LIMITS
23D6	3802	0766		JR C,NUM1 ;JUMP IF NEG.
		0767		;ELSE NUMBER IS >= 'A' (HEX)
23D8	C607	0768		ADD A,07H ;SET UP FOR ASCII
23DA	C630	0769	NUM1	ADD A,30H ;MAKE IT ASCII
23DC	D7	0770		DEFB DISF
23DD	C9	0771		RET
		0772		;
		0773		; CHECK AND SCROLL IF NEEDED
		0774		;
23DE	3ABA4F	0775	SCROLL	LD A,(SCRLN)
23E1	D606	0776		SUB 6 ;SCRLN-6 IS LOWER LIMIT OF SCREEN
23E3	BA	0777		CP D ;D IS Y COORDINATE
23E4	F0	0778		RET P ;GO HOME IF NOT THERE YET
23E5	F5	0779		PUSH AF ;SAVE THIS VALUE
23E6	AF	0780		XOR A ;CLEAR A AND CARRY FLAG
23E7	2AB84F	0781		LD HL,(SCRN) ;GET SCREEN SIZE
23EA	01F000	0782		LD BC,0F0H ;MINUS 1 LINE'S WORTH
23ED	ED42	0783		SBC HL,BC
23EF	280A	0784		JR Z,SCR1 ;IF ZERO, JUST BLANK TOP LINE
23F1	E5	0785		PUSH HL
23F2	C1	0786		POP BC ;HOW MANY TO MOVE
23F3	21F040	0787		LD HL,40F0H ;SOURCE
23F6	110040	0788		LD DE,4000H ;DESTINATION
23F9	EDB0	0789		LDIR ;MOVE (HL) TO (DE) BC TIMES
23FB	01A006	0790	SCR1	LD BC,06A0H ;BLACK OUT LAST LINE
23FE	5F	0791		LD E,A
23FF	F1	0792		POP AF ;RESET COORDS
2400	57	0793		LD D,A
2401	AF	0794		XOR A ;DATA FOR RECTANGLE
2402	FF	0795		DEFB SYSTEM
2403	1C	0796		DEFB 1CH ;RECTAN
2404	C9	0797		RET ;GO HOME
		0798		;
2405	ED5BC34F	0799	CRLF	LD DE,(COORD) ;GET CURSOR COORDS
2409	010305	0800		LD BC,0503H ;PAINT 3X5 BLANK
240C	3E00	0801		LD A,00H ;COLOR 0
240E	FF	0802		DEFB SYSTEM
240F	1C	0803		DEFB 1CH ;RECTAN
		0804		;UPDATE COORDINATES
2410	1E00	0805	CLF	LD E,0 ;'CARRIAGE RETURN'
2412	3E06	0806		LD A,6H ;'LINE FEED'
2414	82	0807		ADD A,D
2415	57	0808		LD D,A
2416	CDDE23	0809		CALL SCROLL
2419	CDA923	0810		CALL CON ;PAINT CURSOR
241C	21AE4F	0811		LD HL,10B
241F	CB46	0812		BIT 0,(HL) ;TAPE WRITE IN PROGRESS?

ADDR	CODE	STMT	SOURCE	STATEMENT
*2421	C8	0813	RET	Z ;NOPE, GO HOME
*2	FF	0814	DEFB	SYSSUK
*2423	51	0815	DEFB	51H ;PAWS
*2424	08	0816	DEFB	8 ;4 CHARACTERS' WORTH
*2425	C9	0817	RET	;AND GO HOME
*2426	C9	0818	RET	
		0819		;
		0820		;THIS ROUTINE OUTPUTS ANY 8 BIT VALUE IN THE A REGISTER
		0821		;TO THE TAPE OUTPUT PORT.
		0822		;NOTE: THE TAPE PORT IS THE SAME AS THE PRINTER PORT
		0823		;SO IT IS SUGGESTED THAT FOR PRINTING THE PRINT COMMAND
		0824		;BE USED SO THAT THE PRINTER DOESN'T GO BANANAS.
		0825		;
*2427		0826	TWRT	;WRITE TO TAPE
*2427	4F	0827	LD	C,A ;CHAR. TO C
*2428	CB01	0828	RLC	C ;SHIFT LEFT ONCE
*242A	DB12	0829	TW1 IN	A,(12H) ;TAPE PORT FEEDBACK
*242C	E602	0830	AND	2 ;WAIT FOR CLOCK HIGH
*242E	28FA	0831	JR	Z,TW1
*2430	060A	0832	LD	B,0AH ;BIT COUNTER
*2432	3EC0	0833	TW2 LD	A,0C0H ;TIME COUNTER (192 DECIMAL)
*2434	3D	0834	TW3 DEC	A
*2435	20FD	0835	JR	NZ,TW3 ;1.72 MSEC., HALF OF 300 HZ
		0836		;CLOCK NOW LOW
*2437	05	0837	DEC	B ;START COUNTING BITS
*2438	C8	0838	RET	Z ;GO HOME IF ALL DONE
*2	DB12	0839	IN	A,(12H) ;IS FEEDBACK STILL SAME STATE?
*243B	5F	0840	LD	E,A ;IF SO, THEN CLOCK IS LOW
*243C	DB12	0841	TW4 IN	A,(12H)
*243E	AB	0842	XOR	E ;IF SAME, RESULT IS 0
*243F	E602	0843	AND	2 ;ELSE RESULT IS 2
*2441	28F9	0844	JR	Z,TW4 ;TRY AGAIN
		0845		;CLOCK HIGH
*2443	7B	0846	LD	A,E ;OLD BIT TO A
*2444	A9	0847	XOR	C ;SEE IF NEED TO CHANGE STATE
*2445	E602	0848	AND	2 ;OF OUTPUT
*2447	2802	0849	JR	Z,TW5 ;NO, SKIP AHEAD
*2449	DB12	0850	IN	A,(12H) ;TOGGLE OUTPUT
*244B	CBC9	0851	TW5 SET	1,C ;PUT IN STOP BITS
*244D	CB09	0852	RRC	C ;SET FOR NEXT BIT
*244F	18E1	0853	JR	TW2 ;DO IT AGAIN
		0854		;
		0855		;
		0856		;
		0857		;
		0858		;
*2451	3E20	0859	SPACE LD	A,20H ;ASCII SPACE
*2453	D7	0860	DEFB	DISP
*2454	C9	0861	RET	
		0862		;
		0863		;
*2	CDCB23'	0864	ERR CALL	RED
*245B	DF	0865	DEFB	STRING
*2459	B226'	0866	DEFW	ERM
*245B	C3B022'	0867	JP	MOD0
		0868		;
		0869		;THIS ROUTINE DISPLAYS THE CONTENTS OF THE A REGISTER.
		0870		;THUS ANY BYTE OF DATA CAN BE DISPLAYED BY PUTTING IT

ADDR	CODE	STMT	SOURCE	STATEMENT
		0871		;IN A AND CALLING RGDIS.
245E	F5	0872	RGDIS	PUSH AF ;DISPLAY REGISTER CONTENTS
245F	E6F0	0873		AND OF0H ;MASK OUT LOWER HALF
2461	07	0874		RLCA
2462	07	0875		RLCA
2463	07	0876		RLCA
2464	07	0877		RLCA
2465	CDD423'	0878	CALL	NUMDIS
2468	F1	0879	POP	AF
2469	E60F	0880	AND	OFH ;MASK OFF UPPER HALF
246B	CDD423'	0881	CALL	NUMDIS
246E	C9	0882	RET	
		0883		;
246F	7C	0884	HLIST	LD A,H ;SHOW HL ROUTINE
2470	CD5E24'	0885	CALL	RGDIS
2473	7D	0886	LD	A,L
2474	CD5E24'	0887	CALL	RGDIS
2477	C9	0888	RET	
		0889		;
		0890		;
>2478		0891	READY	
2478	ED5BB84F'	0892	LD	DE,(SCRN) ;HOW MANY SCREEN BYTES
247C	21504F	0893	LD	HL,UPRAM ;1ST SYSTEM SCRATCH BYTE
247F	CBB4	0894	RES	6,H ;MAKE INTO A RELATIVE NUMBER
2481	B7	0895	OR	A ;CLEAR CARRY FLAG
2482	ED52	0896	SBC	HL,DE ; (TOTAL RAM-SCREEN RAM)
2484	CD6F24'	0897	CALL	HLIST ;DISPLAY IT
2487	DF	0898	DEFB	STRING
2488	6326'	0899	DEFW	RMSG ;STRING TO GO ALONG WITH IT
248A	2AB84F'	0900	LD	HL,(SCRN)
248D	CBF4	0901	SET	6,H ;FUDGE IT INTO AN ADDRESS
248F	CD6F24'	0902	CALL	HLIST ;DUMP IT OUT
2492	CD0524'	0903	CALL	CRLF
2495	C9	0904	RET	
		0905		;
		0906		;
		0907	*****KEYBOARD INTERFACE*****	
		0908		;
		0909	;SCANS THE KEYPAD UNTIL A KEY IS PRESSED. RETURNS WITH	
		0910	;THE NUMBER OF THE KEY IN B. NUMBERS ARE ROW-WISE	
		0911	;RIGHT TO LEFT, 1 THROUGH 24.	
		0912		;
2496	219624'	0913	KEYGET	LD HL,KEYGET ;SAVE THIS ADDRESS
2499	E5	0914		PUSH HL
249A	FF	0915	DEFB	SYSSUK ;
249B	51	0916	DEFB	51H ;PAWS--KEY DEBOUNCER, HOLD FOR
249C	02	0917	DEFB	02H ;2/60 OF A SECOND
		0918	;KEYPAD INPUT RETURNS WITH KEY NO. IN A	
249D	11AF4F'	0919	LD	DE,KEYN ;OLD KEY NUMBER
24A0	011404	0920	LD	BC,414H ;B=COUNT, C=STARTING PORT
24A3	ED78	0921	KG1	IN A,(C) ;CHECK OUT PORT
24A5	E63F	0922	AND	3FH ;GET RID OF EXTRANEIOUS BITS
24A7	2006	0923	JR	NZ,KG2 ;JUMP IF GOT A GOOD ONE
24A9	0C	0924	INC	C ;NOPE, DO ANOTHER
24AA	10F7	0925	DJNZ	KG1
24AC	AF	0926	XOR	A ;NONE AT ALL
24AD	12	0927	LD	(DE),A ;KEEP TRACK OF LAST KEY
24AE	C9	0928	RET	;BACK TO KEYGET

ADDR	CODE	STMT	SOURCE	STATEMENT
		0929	;	
'24	05	0930	KG2	DEC B
'24B0	0E00	0931		LD C,0
'24B2	0F	0932	KG3	RRCA ;FIND THE RIGHT BIT
'24B3	3B03	0933		JR C,KG4
'24B5	0C	0934		INC C
'24B6	18FA	0935		JR KG3
'24B8	79	0936	KG4	LD A,C ;BIT #
'24B9	07	0937		RLCA ;MULT BY 4
'24BA	07	0938		RLCA
'24BB	B0	0939		OR B
'24BC	3C	0940		INC A ;NOW HAVE KEY NO.
'24BD	47	0941		LD B,A
'24BE	1A	0942		LD A,(DE) ;OLD NUMBER
'24BF	A8	0943		XOR B ;COMPARE
'24C0	C8	0944		RET Z
'24C1	78	0945		LD A,B ;DIFFERENT, UPDATE
'24C2	12	0946		LD (DE),A
'24C3	E1	0947		POP HL ;FIX THE STACK
'24C4	C9	0948		RET ;AND GO BACK
		0949	;	
		0950	*****INTERRUPT ROUTINES*****	
		0951	;	
		0952	;	
		0953	;INITIALIZE INTERRUPTS	
		0954	;	
'24	DB12	0955	ININT	IN A,(12H) ;TAPE INPUT PORT
'24C7	E602	0956		AND 2 ;MAKE SURE IT'S SET TO 0
'24C9	20FA	0957		JR NZ,ININT
'24CB	F3	0958		DI ;DISABLE INTERRUPTS
'24CC	ED5E	0959		IM 2 ;INTERRUPT MODE
'24CE	3E4F	0960		LD A,4FH ;SET INTERRUPT PAGE
'24D0	ED47	0961		LD I,A
'24D2	3EB2	0962		LD A,0B2H ;SCREEN INTERRUPT VECTOR
'24D4	D30D	0963		OUT (0DH),A
'24D6	3E08	0964		LD A,08
'24D8	D30E	0965		OUT (0EH),A ;SCREEN INTS ONLY
'24DA	3EC8	0966		LD A,200
'24DC	D30F	0967		OUT (0FH),A ;SCREEN INT. EVERY 200 LINES
'24DE	21EC24'	0968		LD HL,TAPINT ;LOAD INTERRUPT VECTORS
'24E1	22B04F'	0969		LD (LPINT),HL
'24E4	216C25'	0970		LD HL,SCRINT
'24E7	22B24F'	0971		LD (SCINT),HL
'24EA	FB	0972		EI ;ENABLE INTERRUPTS
'24EB	C9	0973		RET
		0974	;	
		0975	*****LIGHT PEN INTERRUPT HANDLER*****	
		0976	;	
		0977	;THE TAPE INPUT HANDLER RECOGNIZES A COLON AS AN	
		0978	;ADDRESS DIRECTIVE, AND SPACE AND CR AS WRITE DIRECTIVES.	
		0979	;	
'24	C	0980	TAPINT	
'24ED	F5	0981		PUSH AF
'24ED	D9	0982		EXX
'24EE	CD4A25'	0983		CALL COLL ;GET A CHARACTER
'24F1	21AE4F'	0984	TI1	LD HL,IOB ;TEST TAPE DISPLAY BIT
'24F4	CB5E	0985		BIT 3,(HL)
'24F6	2045	0986		JR NZ,TID ;IF SET, GO DISPLAY

ADDR	CODE	STMT	SOURCE	STATEMENT
'24FB	FE0D	0987	CP	ODH ;ELSE TEST FOR CR
'24FA	2007	0988	JR	NZ,TI2 ;SKIP IF NOT
'24FC	CD3321'	0989	CALL	WRITE ;DO A WRITE
'24FF	3E0D	0990	LD	A,ODH ;REPLACE CR
'2501	183A	0991	JR	TID ;DISPLAY IT
'2503	FE20	0992	CP	20H ;ELSE, ACT ON IT
'2505	3836	0993	JR	C,TID ;EXIT IF NOT DISPLAYABLE
'2507	200B	0994	JR	NZ,TI3 ;ZERO MEANS A SPACE
'2509	CB7E	0995	BIT	7,(HL)
		0996	;BIT 7 IN IOB IS ADDR FLAG MEANING IGNORE 1ST SPACE	
		0997	;FOLLOWING COLON	
'250B	E5	0998	PUSH	HL ;SAVE IOB
'250C	CC3321'	0999	CALL	Z,WRITE ;IF CLEAR
'250F	E1	1000	POP	HL
'2510	CBBE	1001	RES	7,(HL) ;CLEAR THE BIT
'2512	182A	1002	JR	TIX ;EXIT
		1003	;	
'2514	CBBE	1004	RES	7,(HL) ;CLEAR BIT IF NOT A SPACE
'2516	FE30	1005	CP	30H ;CHECK NUMBER LIMITS
'2518	3823	1006	JR	C,TID ;LESS THAN
'251A	FE3A	1007	CP	3AH
'251C	3008	1008	JR	NC,TI4 ;>=
'251E	E60F	1009	AND	0FH ;STRIP AWAY ASCII
'2520	4F	1010	LD	C,A
'2521	CDF220'	1011	CALL	NUMBER
'2524	1818	1012	JR	TIX
'2526	2009	1013	JR	NZ,TI5 ;JUMP IF NOT A COLON
'2528	E5	1014	PUSH	HL ;SAVE IOB
'2529	CD0E21'	1015	CALL	ADDR
'252C	E1	1016	POP	HL
'252D	CBFE	1017	SET	7,(HL) ;IGNORE NEXT SPACE
'252F	180D	1018	JR	TIX ;EXIT
'2531	FE41	1019	CP	41H ;'A'
'2533	3808	1020	JR	C,TID
'2535	FE47	1021	CP	47H ;'G'
'2537	3004	1022	JR	NC,TID
'2539	C609	1023	ADD	A,09H ;SET UP FOR NUMBER
'253B	18E1	1024	JR	TI3A ;GO TO IT
'253D	D7	1025	DEFB	DISP
'253E	3AAD4F'	1026	LD	A,(MODE)
'2541	FE04	1027	CP	4 ;MODE 4 SET?
'2543	CA4323'	1028	JP	Z,MODE4 ;SKIP OUT
'2546	F1	1029	POP	AF
'2547	D9	1030	EXX	
'2548	FB	1031	EI	
'2549	C9	1032	RET	
		1033	;	
'254A	DB12	1034	COLL	IN A,(12H) ;COLLECT BITS FROM TAPE
'254C	1F	1035	RRA	;BIT TO CARRY
'254D	79	1036	LD	A,C ;BITS SO FAR
'254E	1F	1037	RRA	;NEW BIT SHIFTED IN
'254F	4F	1038	LD	C,A
'2550	78	1039	LD	A,B ;COUNT
'2551	A7	1040	AND	A
'2552	380A	1041	JR	C,C01 ;NEG.
'2554	2012	1042	JR	NZ,C02 ;NONZERO
'2556	CB79	1043	BIT	7,C ;ZERO, TEST HIGHEST BIT
'2558	200B	1044	JR	NZ,EXIT ;NOT READY YET

ADDR	CODE	STMT	SOURCE	STATEMENT
*255A	0608	1045	LD	B,08H ; GOT ONE
*255B	1807	1046	JR	EXIT
*255C	04	1047	CO1 INC	B ; NEG. COUNT LOOP
*255F	CB79	1048	BIT	7,C ; STILL COLLECTING BITS
*2561	2002	1049	JR	NZ,EXIT
*2563	06FC	1050	LD	B,0FCH ; A ZERO CAME THRU WITH
		1051		; COUNT WRONG
*2565	E1	1052	EXIT POP	HL ; DROP RETURN ADDRESS
*2566	18D6	1053	JR	TIX ; AND GET OUT
		1054		;
*2568	10FB	1055	CO2 DJNZ	EXIT ; POS COUNT, NOT DONE YET
*256A	79	1056	LD	A,C ; DONE
*256B	C9	1057	RET	
		1058		;
		1059	*****SCREEN INTERRUPT HANDLER*****	
		1060		;
*256C	FB	1061	SCRINT EI	; ENABLE INTERRUPTS
*256D	C9	1062	RET	; AND GO BACK
		1063		;
		1064		;
		1065		;
		1066	*****UTILITY PROGRAMS*****	
		1067		;
		1068		;
		1069	; BREAKPOINTS ARE SET BY WRITING 'CF' AT DESIRED LOCATIONS	
		1070		;
*256E	CD9A21	1071	BRKPT CALL	CART ; BREAKPOINT ROUTINE
*256F	CD0524	1072	CALL	CRLF ; CART GETS ORIGINAL REG SET
*2574	DF	1073	DEFB	STRING
*2575	4126	1074	DEFW	BKM
*2577	E1	1075	POP	HL ; GET BRKPT ADDR
*2578	2B	1076	DEC	HL ; BACK OFF ONE
*2579	CD6F24	1077	CALL	HLIST ; SHOW IT
*257C	CD7322	1078	CALL	RLIST ; SHOW REGISTERS
*257F	C30020	1079	JP	START
		1080		;
		1081		;
		1082		;
		1083	; SET SCREEN SPECIFICATIONS	
		1084		;
*2582	CD0524	1085	SCRSP CALL	CRLF ; OUTPUT A CARRIAGE RETURN
*2585	3E5A	1086	LD	A,90 ; DEFAULT VALUE
*2587	32B44F	1087	LD	(IN1),A ; TO INPUT REGISTER
*258A	DF	1088	DEFB	STRING
*258B	8126	1089	DEFW	BLM ; BLANK SPECS MESSAGE
*258D	AF	1090	XOR	A ; USE REG TYPE SEQUENCE
*258E	32BD4F	1091	LD	(RFG),A
*2591	3E03	1092	LD	A,3
*2593	32AD4F	1093	LD	(MODE),A ; SET MODE 3
*2596	C9	1094	RET	
		1095		;
		1096	; THE FOLLOWING SECTION WILL BE ENTERED EACH TIME A 'W' IS	
		1097	; INPUT WHILE IN MODE 3	
		1098		;
*2597	79	1099	MODE3 LD	A,C ; GET KEY VALUE
*2598	FE04	1100	CP	4H
*259A	CO	1101	RET	NZ ; ACCEPT WRITE ONLY
*259B	3ABD4F	1102	LD	(RFG) ; PICK UP FLAG

ADDR	CODE	STMT	SOURCE STATEMENT
*259E	FF	1103	DEFB SYSSUK
*259F	5B	1104	DEFB 5BH ; INDEXW-JUMP BASED ON FLAG
*25A0	EA20	1105	DEFW M3JT ; JUMP TABLE
*25A2	D5	1106	PUSH DE ; RETURNED ADDRESS
*25A3	3AB44F	1107	LD A,(IN1) ; GET INPUT VALUE
*25A6	C9	1108	RET ; JUMP TO ROUTINE
		1109 ;	
*25A7		1110 M30	; CALCULATE WHERE BLANKING
		1111	; SHOULD START
*25A7	FE5A	1112	CP 5AH ; IF 90 DECIMAL
*25A9	280B	1113	JR Z,M302 ; LEAVE ALONE
*25AB	E60F	1114	AND 0FH ; MASK OUT UPPER DIGIT
*25AD	2001	1115	JR NZ,M300 ; IF ZERO, MAKE IT ONE
*25AF	3C	1116	INC A
*25B0	47	1117 M300	LD B,A ; NO. OF TEXT LINES INPUT
*25B1	AF	1118	XOR A
*25B2	C606	1119 M301	ADD A,06H ; CHARACTER FRAME SIZE
*25B4	10FC	1120	DJNZ M301
*25B6	32B64F	1121 M302	LD (ADRG1),A ; TEMP. LINE STORAGE
*25B9	3E2C	1122	LD A,2CH ; 44 DECIMAL
*25BB	32B44F	1123	LD (IN1),A ; HORIZONTAL COLOR BOUNDARY
*25BE	CD0524	1124	CALL CRLF
*25C1	DF	1125	DEFB STRING
*25C2	8E26	1126	DEFW CBM
*25C4	3E01	1127	LD A,1 ; INCREMENT PROCESS POINTER
*25C6	32BD4F	1128	LD (RFG),A
*25C9	C9	1129	RET ; GET THE NEXT SPEC.
		1130 ;	
*25CA	32BB4F	1131 M31	LD (HRZCB),A ; NO LIMIT CHECKS
*25CD	3E00	1132	LD A,00H ; DEFAULT BACK COLOR
*25CF	32B44F	1133	LD (IN1),A
*25D2	CD0524	1134	CALL CRLF
*25D5	DF	1135	DEFB STRING
*25D6	9F26	1136	DEFW BCLM ; BACK COLOR MESSAGE
*25D8	3E02	1137	LD A,2 ; SET UP FOR NEXT PROCESS
*25DA	32BD4F	1138	LD (RFG),A
*25DD	C9	1139	RET ; GO BACK
		1140 ;	
*25DE	32C94F	1141 M32	LD (BCOL),A ; SET BCOLOR
*25E1	3E07	1142	LD A,07H ; FORE COLOR
*25E3	32B44F	1143	LD (IN1),A
*25E6	CD0524	1144	CALL CRLF
*25E9	DF	1145	DEFB STRING
*25EA	4C26	1146	DEFW FCLM ; FORE COLOR MESSAGE
*25EC	3E03	1147	LD A,3 ; SET UP FOR NEXT PROCESS
*25EE	32BD4F	1148	LD (RFG),A
*25F1	C9	1149	RET
		1150 ;	THIS IS THE ACTION OF THE LAST KEYPUSH OF THE
		1151 ;	SCREEN SPEC. ROUTINE
*25F2	32C84F	1152 M33	LD (FCOL),A ; SET FCOLOR
*25F5	3AB64F	1153	LD A,(ADRG1) ; GET BLANK LINE #
*25F8	32BA4F	1154	LD (SCRLN),A ; INTO ITS PROPER PLACE
*25FB	07	1155	RLCA ; SETOUT NEEDS IT
		1156	; SHIFTED LEFT ONE BIT
*25FC	57	1157	LD D,A
*25FD	3ABB4F	1158	LD A,(HRZCB) ; COLOR BOUNDARY
*2600	47	1159	LD B,A
*2601	3E08	1160	LD A,08H ; INTERRUPT MODE

ADDR	CODE	STMT	SOURCE	STATEMENT
'2603	FF	1161	DEFB	SYSTEM
'2604	16	1162	DEFB	16H ;DO NT SETOUT
'2605	21C64F'	1163	LD	HL,COLORS ;GET ADDRESS OF COLOR LIST
'2608	FF	1164	DEFB	SYSTEM
'2609	18	1165	DEFB	18H ;COLSET
'260A	210000	1166	LD	HL,00H ;CALCULATE SCREEN SIZE
		1167		;FROM NO. OF LINES DISPLAYED
'260D	3ABA4F'	1168	LD	A,(SCRLN)
'2610	47	1169	LD	B,A
'2611	112800	1170	LD	DE,40 ;DECIMAL BYTES PER LINE
'>2614		1171	M331	
'2614	19	1172	ADD	HL,DE
'2615	10FD	1173	DJNZ	M331 ;FIGURE OUT
'2617	22B84F'	1174	LD	(SCRN),HL ;NO. OF BYTES IN SCREEN
'261A	CD4723'	1175	CALL	CLEAR ;CLEAR SCREEN
'261D	CD7824'	1176	CALL	READY
'2620	AF	1177	XOR	A ;CLEAN UP
'2621	32BD4F'	1178	LD	(RFG),A
'2624	32AD4F'	1179	LD	(MODE),A
'2627	C9	1180	RET	;GO HOME
		1181	*****MESSAGE	TABLE*****
'2628	413A	1182	AFM	DEFM 'A:'
'262A	00	1183	DEFB	00
'262B	2042433A	1184	BCM	DEFM 'BC:'
'262F	00	1185	DEFB	00
'2630	2044453A	1186	DEM	DEFM 'DE:'
'2631	00	1187	DEFB	00
'2635	20484C3A	1188	HLM	DEFM 'HL:'
'2639	00	1189	DEFB	00
'263A	20414444	1190	ADM	DEFM 'ADDR:'
	523A			
'2640	00	1191	DEFB	00
'2641	424B5054	1192	BKM	DEFM 'BKPT ADDR:'
	20414444			
	523A			
'264B	00	1193	DEFB	00
'264C	464F5245	1194	FCLM	DEFM 'BACKGROUND:'
	47524F55			
	4E443A20			
'2658	00	1195	DEFB	00
'2659	2A574152	1196	WAM	DEFM '*WARNING*'
	4E494E47			
	2A			
'2662	00	1197	DEFB	00
'2663	20425954	1198	RMSG	DEFM 'BYTES AVAILABLE STARTING AT '
	45532041			
	5641494C			
	41424C45			
	20535441			
	5254494E			
	47204154			
	20			
'2660	00	1199	DEFB	00
'2681	54455854	1200	BLM	DEFM 'TEXT LINES:'
	204C494E			
	45533A20			
'268D	00	1201	DEFB	00
'268E	434F4C4F	1202	CBM	DEFM 'COLOR BOUNDARY:'

ADDR	CODE	STMT	SOURCE	STATEMENT
	5220424F			
	554E4441			
	52593A20			
*269E	00	1203	DEFB	00
*269F	4241434B	1204	DEFM	'BACKGROUND COLOR: '
	47524F55			
	4E442043			
	4F4C4F52			
	3A20			
*26B1	00	1205	DEFB	00
*26B2	455252	1206	DEFM	'ERR'
*26B5	00	1207	DEFB	00
*26B6	4F4B	1208	DEFM	'OK'
*26B8	00	1209	DEFB	00
		1210		;
		1211		;
		1212	*****COLOR LIST*****	
*26B9	A3	1213	COLIST	DEFB 0A3H ; GREEN
*26BA	52	1214		DEFB 52H ; RED
*26BB	07	1215		DEFB 07H ; WHITE
*26BC	00	1216		DEFB 00H ; BLACK
		1217		;
		1218		;
		1219	*****SMALL CHARACTER FONT DESCRIPTOR*****	
*26BD	A0	1220	SMLFNT	DEFB 0A0H ; 1ST CHARACTER
*26BE	04	1221		DEFB 4 ; X FRAME SIZE
*26BF	06	1222		DEFB 6 ; Y FRAME SIZE
*26C0	01	1223		DEFB 1 ; NO. OF BYTES X PATTERN
*26C1	05	1224		DEFB 5 ; NO. OF BYTES Y PATTERN
*26C2	C426'	1225		DEFW SMLCHR ; ADDRESS OF CHARACTER TABLE
		1226		;
		1227		;
		1228	*****CHARACTER TABLE*****	
*26C4	00000000	1229	SMLCHR	DEFB 00,00,00,00,00 ; SPACE
	00			
*26C9	20202000	1230		DEFB 20H,20H,20H,00,20H ; !
	20			
*26CE	A0A00000	1231		DEFB 0A0H,0A0H,00,00,00 ; "
	00			
*26D3	A0E0A0E0	1232		DEFB 0A0H,0E0H,0A0H,0E0H,0A0H ; #
	A0			
*26D8	00000000	1233		DEFB 0,0,0,0,0 ; UNDEFINED
	00			
*26DD	A0204080	1234		DEFB 0A0H,20H,40H,80H,0A0H ; %
	A0			
*26E2	00000000	1235		DEFB 0,0,0,0,0 ; UNDEFINED
	00			
*26E7	40800000	1236		DEFB 40H,80H,0,0,0 ; '
	00			
*26EC	40808080	1237		DEFB 40H,80H,80H,80H,40H ; (
	40			
*26F1	40202020	1238		DEFB 40H,20H,20H,20H,40H ;)
	40			
*26F6	00A040A0	1239		DEFB 00H,0A0H,40H,0A0H,00H ; *
	00			
*26FB	0040E040	1240		DEFB 00H,40H,0E0H,40H,00H ; +
	00			
*2700	00000040	1241		DEFB 00H,00H,00H,40H,80H ; ,

ADDR	CODE	STMT	SOURCE	STATEMENT
	80			
*2	0000E000	1242	DEFB	00H, 00H, 0E0H, 00H, 00H ; -
	00			
*270A	00000000	1243	DEFB	00H, 00H, 00H, 00H, 40H ; .
	40			
*270F	20204080	1244	DEFB	20H, 20H, 40H, 80H, 80H ; /
	80			
*2714	40A0A0A0	1245	DEFB	40H, 0A0H, 0A0H, 0A0H, 40H ; 0
	40			
*2719	40404040	1246	DEFB	40H, 40H, 40H, 40H, 40H ; 1
	40			
*271E	E020E080	1247	DEFB	0E0H, 20H, 0E0H, 80H, 0E0H ; 2
	E0			
*2723	E0206020	1248	DEFB	0E0H, 20H, 60H, 20H, 0E0H ; 3
	E0			
*2728	A0A0E020	1249	DEFB	0A0H, 0A0H, 0E0H, 20H, 20H ; 4
	20			
*272D	E080E020	1250	DEFB	0E0H, 80H, 0E0H, 20H, 0E0H ; 5
	E0			
*2732	E080E0A0	1251	DEFB	0E0H, 80H, 0E0H, 0A0H, 0E0H ; 6
	E0			
*2737	E0202020	1252	DEFB	0E0H, 20H, 20H, 20H, 20H ; 7
	20			
*273C	E0A0E0A0	1253	DEFB	0E0H, 0A0H, 0E0H, 0A0H, 0E0H ; 8
	E0			
*2741	E0A0E020	1254	DEFB	0E0H, 0A0H, 0E0H, 20H, 20H ; 9
	20			
*2746	00400040	1255	DEFB	00H, 40H, 00H, 40H, 00H ; :
	00			
*274B	00400040	1256	DEFB	00H, 40H, 00H, 40H, 80H ; ;
	80			
*2750	20408040	1257	DEFB	20H, 40H, 80H, 40H, 20H ; <
	20			
*2755	00E000E0	1258	DEFB	00H, 0E0H, 00H, 0E0H, 00H ; =
	00			
*275A	80402040	1259	DEFB	80H, 40H, 20H, 40H, 80H ; >
	80			
*275F	E0206040	1260	DEFB	0E0H, 20H, 60H, 40H, 40H ; ?
	40			
*2764	E0A020E0	1261	DEFB	0E0H, 0A0H, 20H, 0E0H, 0E0H ; @
	E0			
*2769	40A0E0A0	1262	DEFB	40H, 0A0H, 0E0H, 0A0H, 0A0H ; A
	A0			
*276E	C0A0C0A0	1263	DEFB	0C0H, 0A0H, 0C0H, 0A0H, 0C0H ; B
	C0			
*2773	E0808080	1264	DEFB	0E0H, 80H, 80H, 80H, 0E0H ; C
	E0			
*2778	C0A0A0A0	1265	DEFB	0C0H, 0A0H, 0A0H, 0A0H, 0C0H ; D
	C0			
*277D	E080C080	1266	DEFB	0E0H, 80H, 0C0H, 80H, 0E0H ; E
	E0			
*2	E080C080	1267	DEFB	0E0H, 80H, 0C0H, 80H, 80H ; F
	80			
*2787	E080A0A0	1268	DEFB	0E0H, 80H, 0A0H, 0A0H, 0E0H ; G
	E0			
*278C	A0A0E0A0	1269	DEFB	0A0H, 0A0H, 0E0H, 0A0H, 0A0H ; H
	A0			
*2791	40404040	1270	DEFB	40H, 40H, 40H, 40H, 40H ; I

ADDR	CODE	STMT	SOURCE STATEMENT
*2796	40 202020A0 E0	1271	DEFB 20H,20H,20H,0A0H,0E0H ; J
*279B	A0A0C0A0 A0	1272	DEFB 0A0H,0A0H,0C0H,0A0H,0A0H ; K
*27A0	B0B080B0 E0	1273	DEFB 80H,80H,80H,80H,0E0H ; L
*27A5	A0E0A0A0 A0	1274	DEFB 0A0H,0E0H,0A0H,0A0H,0A0H ; M
*27AA	B0E0E0A0 A0	1275	DEFB 80H,0E0H,0E0H,0A0H,0A0H ; N
*27AF	E0A0A0A0 E0	1276	DEFB 0E0H,0A0H,0A0H,0A0H,0E0H ; O
*27B4	E0A0E080 80	1277	DEFB 0E0H,0A0H,0E0H,80H,80H ; P
*27B9	E0A0A0E0 20	1278	DEFB 0E0H,0A0H,0A0H,0E0H,20H ; Q
*27BE	E0A0C0A0 A0	1279	DEFB 0E0H,0A0H,0C0H,0A0H,0A0H ; R
*27C3	60B0E020 C0	1280	DEFB 60H,80H,0E0H,20H,0C0H ; S
*27C8	E0404040 40	1281	DEFB 0E0H,40H,40H,40H,40H ; T
*27CD	A0A0A0A0 E0	1282	DEFB 0A0H,0A0H,0A0H,0A0H,0E0H ; U
*27D2	A0A0A0A0 40	1283	DEFB 0A0H,0A0H,0A0H,0A0H,40H ; V
*27D7	A0A0A0E0 A0	1284	DEFB 0A0H,0A0H,0A0H,0E0H,0A0H ; W
*27DC	A0A040A0 A0	1285	DEFB 0A0H,0A0H,40H,0A0H,0A0H ; X
*27E1	A0A04040 40	1286	DEFB 0A0H,0A0H,40H,40H,40H ; Y
*27E6	E0204080 E0	1287	DEFB 0E0H,20H,40H,80H,0E0H ; Z
*27EB	28432920 31393831 20412E20 47554556 415241	1288 ; 1289	DEFM '(C) 1981 A. GUEVARA'
*27FE	00	1290 1291 ; 1292	DEFB 00 END

ADDR CODE STMT SOURCE STATEMENT

C. S REFERENCE LISTING

SYMBOL	VALUE	TYPE	STMT	STATEMENT REFERENCES							
ACALL	218F		0339	0201	0178						
ADDR	210E		0250	1015	0199	0176					
ADM	263A		1190	0527							
ADRG1	4FB6		0029	1153	1121	0628	0528	0397	0388	0378	0311
-				0304	0279	0275	0268	0265	0257		
ADRG2	4FB7		0030								
AFG	2224		0458	0443							
AFM	2628		1182	0511	0500	0435					
BCG	222C		0462	0444							
BCLM	269F		1204	1136							
BCM	262B		1184	0501							
BCOL	4FC9		0045	1141							
BKM	2641		1192	1074							
BLM	2681		1200	1089							
BPE	2007		0051								
BRKPT	256E		1071	0051							
C2	4FC7		0043								
CART	219A		0351	1071	0339						
CBM	268E		1202	1126							
CLEAR	2347		0659	1175	0088						
CLF	2410		0805	0715							
C	255E		1047	1041							
COZ	2568		1055	1042							
COLIST	2689		1213	0080							
COLL	254A		1034	0983							
COLORS	4FC6		0042	1163	0082	0078					
CON	23A9		0723	0810	0716						
CONT	204E		0090	0064							
COORD	4FC3		0039	0799	0726	0700	0666				
CRLF	2405		0799	1144	1134	1124	1085	1072	0903	0690	0570
-				0530	0509	0106					
D0	2370		0690								
D1	2375		0692	0687							
DEG	2238		0469	0445							
DEL	2325		0625	0206							
DEM	2630		1186	0502							
DEP	200A		0052								
DISP	00D7		0013	1025	0860	0770	0743	0639	0619	0608	0593
-				0427	0423	0400	0376	0361	0252		
DISPLA	2357		0675	0052							
DOT	2370		0696	0693							
DRET	23A4		0717	0691							
END	4FC1		0038	0640	0627	0329	0321	0319	0310	0292	
ERM	2682		1206	0866							
ERR	2455		0864	0307							
EXIT	2565		1052	1055	1049	1046	1044				
F	264C		1194	1146							
FCUL	4FC8		0044	1152							
G1	23C6		0751	0761	0757						
GREEN	23C3		0749	0617	0602	0591					
HLG	2244		0476	0446							
HLIST	246F		0884	1077	0902	0897	0539	0529	0398		
HLM	2635		1188	0503							
HR7CR	4F8B		0033	1158	131						

ADDR	CODE	STMT	SOURCE	STATEMENT
IN1	4FB4	0027	1143	1133 1123 1107 1087 0476 0469 0462 0458 0387 0377 0341 0276 0258 0218 0098
IN2	4FB5	0028		
INAD	2119	0257	0250	0099
ININT	24C5	0955	0957	0092
INIT	2010	0054	0048	
INS	2142	0292	0183	
INS1	215A	0303	0297	
INS2	2168	0310	0306	0302 0296
INS3	217E	0321	0316	
I0B	4FAE	0023	0984	0811 0680 0615 0598 0583 0575 0409 0101
KEYGET	2496	0913	0913	0113
KEYN	4FAF	0024	0919	
KG1	24A3	0921	0925	
KG2	24AF	0930	0923	
KG3	24B2	0932	0935	
KG4	24B8	0936	0933	
LA1	21F7	0419	0425	
LA2	21FE	0423	0421	
LA3	2202	0426	0412	
LASC	21F1	0416	0411	
LFLG	4FBC	0034	0547	0396 0371 0254 0104
LIST	21AC	0371	0179	
LPINT	4FBO	0025	0969	
MOJT	20BA	0176	0236	
M1JT	20CA	0188	0243	
M1RT	210D	0245	0195	0194 0193 0190 0188
M2JT	20DA	0199	0557	
M30	25A7	1110	0211	
M300	25B0	1117	1115	
M301	25B2	1119	1120	
M302	25B6	1121	1113	
M31	25CA	1131	0212	
M32	25DE	1141	0213	
M33	25F2	1152	0214	
M331	2614	1171	1173	
M3JT	20EA	0211	1105	
M4BA	4FCA	0046	0653	
MAIN	207D	0110	0110	
MDTBL	20B0	0166	0132	
MLIST	21C0	0384	0390	
MOD0	22B0	0545	0867	0642 0620 0609 0594 0577 0571 0531 0497
MODE	4FAD	0022	1179	1093 1026 0546 0433 0358 0128 0102 0068
MODE0	2100	0233	0166	
MODE1	2107	0240	0167	
MODE2	22BB	0553	0168	
MODE3	2597	1099	0169	
MODE4	2343	0652	1028	0170
NORM	23D0	0759	0362	0301
NUM1	23DA	0769	0766	
NUMBER	20F2	0218	1011	0127
NUMDIS	23D4	0765	0881	0878 0229
OK	237F	0697	0695	
OK1	2392	0706	0704	

ADDR	CODE	STMT	SOURCE	STATEMENT										
DI	26B6	1208	0108											
OL1	21E0	0402	0407											
OUTLN	21CF	0395	0384	0380	0373									
POPT	4FC5	0041	0751	0701	0548	0103								
PR1	2306	0604	0600											
PR2	230D	0607	0603											
PRINT	22F8	0597	0202											
PW2	4FC0	0037	0096	0062										
PWRUP	4FBE	0036	0094	0056										
READ	2124	0265	0402	0177										
READY	2478	0891	1176	0089										
RED	23CB	0755	0864	0638	0604	0588	0359	0298						
REG	2206	0432	0181											
RFG	4FBD	0035	1178	1148	1138	1128	1102	1091	0508	0496				
			0489	0485	0482	0452	0437	0105						
RGDIS	245E	0872	0887	0885	0516	0269								
RGTBL	2213	0443	0455											
RLIST	2273	0507	1078	0483	0191									
RMSG	2663	1198	0899											
RMTBL	226B	0500	0492											
RP1	2264	0495	0487											
RPLUS	2251	0485	0540	0517	0474	0467	0460	0189						
RWRT	221B	0452	0192											
SCINT	4FB2	0026	0971											
SCPL	23FB	0790	0784											
SI VT	256C	1061	0970											
SCRLN	4FBA	0032	1168	1154	0775	0087								
SCRN	4FB8	0031	1174	0900	0892	0781	0660	0085						
SCROLL	23DE	0775	0809											
SCRSP	2582	1085	0097											
SEP	200D	0053												
SKYD	0013	0012												
SMLCHR	26C4	1229	1225											
SMLFNT	26BD	1220	0708											
SPACE	2451	0859	0401	0280	0270									
STAR0	219D	0357	0182											
STAR2	2313	0612	0205											
START	2000	0048	1079											
STR1	23BB	0740	0745	0493										
STRDIS	23B5	0733	0053											
STRING	00DF	0014	1145	1135	1125	1088	1073	0898	0865	0526				
			0510	0434	0299	0107								
STSPC	2026	0065	0060											
SWDIS	22A4	0535	0524	0522	0520									
SYSSUK	00FF	0015	1103	0915	0814	0555	0490	0453	0241	0234				
			0130	0116	0090	0065								
SYSTEM	00FF	0016	1164	1161	0802	0795	0727	0709	0663					
TADIS	22D2	0573	0204											
TAPIN	22C2	0562	0574	0200										
TAPINT	24EC	0980	0968											
TI JT	22DD	0582	0203											
TI1	24F1	0984												
TI2	2503	0992	0988											
TI3	2514	1004	0994											
TI3A	251E	1009	1024											
TI4	2526	1013	1008											
TI5	2531	1019	1013											
TID	253D	1025	101	1020	1006	0993	0991	0986						

ADDR	CODE	STMT	SOURCE	STATEMENT
TIX	253E	1026	1053	1018 1012 1002
T01	22ED	0590	0585	
T02	22F2	0592	0589	
TTT	2098	0139	0118	
TW1	242A	0829	0831	
TW2	2432	0833	0853	
TW3	2434	0834	0835	
TW4	243C	0841	0844	
TW5	244B	0851	0849	
TWRT	2427	0826	0699 0689	0682
UPRAM	4F50	0017	0893 0293	
UPSTK	4FAC	0021	0055	
WAM	2659	1196	0300	
WRITE	2133	0275	0999 0989	0328 0180

ERRORS=0000

APPENDIX B:

Z-80 Instruction Set

ADDR	CODE	STMT	SOURCE	STATEMENT
		0002	;	PSEUDO OPS
		0003	;	
>0000		0004	ORG	0 ; ORIGIN (STARTING ADDRESS)
		0005	;	
*0000	AA	0006	DEFB	0AAH ; DEFINE BYTE
*0001	BBAA	0007	DEFW	0AABBH ; DEFINE WORD
*0003	41424344	0008	DEFM	'ABCD' ; DEFINE MESSAGE
>0007		0009	NN	DEFS 2 ; DEFINE STORAGE (2 BYTES)
>0005		0010	IND	EQU 5 ; DISPLACEMENT (IX,IY)
>0020		0011	N	EQU 20H ; AN IMMEDIATE VALUE
>0030		0012	DIS	EQU 30H ; DISPLACEMENT FOR RELATIVE JUMPS
		0013	;	
		0014	;	
		0015	;	
		0016	;	Z80 OPCODES
		0017	;	
*0009	8E	0018	ADC	A,(HL) ; ADD BYTE AT (HL) TO A W/CARRY
*000A	DD8E05	0019	ADC	A,(IX+IND)
*000D	FD8E05	0020	ADC	A,(IY+IND) ; IND=05 FOR ASS'Y PURPOSES
*0010	8F	0021	ADC	A,A
*0011	88	0022	ADC	A,B
*0012	89	0023	ADC	A,C
*0013	8A	0024	ADC	A,D
*0014	8B	0025	ADC	A,E
*0015	8C	0026	ADC	A,H
*0016	8D	0027	ADC	A,L
*0017	CE20	0028	ADC	A,N ; N=20H FOR ASSEMBLY
*0018	ED4A	0029	ADC	HL,BC ; ADD HL TO BC W/CARRY
*001B	ED5A	0030	ADC	HL,DE
*001D	ED6A	0031	ADC	HL,HL
*001F	ED7A	0032	ADC	HL,SP
		0033	;	
*0021	86	0034	ADD	A,(HL)
*0022	DD8605	0035	ADD	A,(IX+IND)
*0025	FD8605	0036	ADD	A,(IY+IND)
*0028	87	0037	ADD	A,A
*0029	80	0038	ADD	A,B
*002A	81	0039	ADD	A,C
*002B	82	0040	ADD	A,D
*002C	83	0041	ADD	A,E
*002D	84	0042	ADD	A,H
*002E	85	0043	ADD	A,L
*002F	C620	0044	ADD	A,N
*0031	09	0045	ADD	HL,BC
*0032	19	0046	ADD	HL,DE
*0033	29	0047	ADD	HL,HL
*0034	39	0048	ADD	HL,SP
*0035	DD09	0049	ADD	IX,BC
*0037	DD19	0050	ADD	IX,DE
*0039	DD29	0051	ADD	IX,IX
*003B	DD39	0052	ADD	IX,SP
*003D	FD09	0053	ADD	IY,BC
*003F	FD19	0054	ADD	IY,DE
*0041	FD29	0055	ADD	IY,IY
*0043	FD39	0056	ADD	IY,SP
		0057	;	
*0045	A6	0058	AND	(HL) ; LOGICAL 'AND' A AND
		0059		; BYTE ADDR BY HL

ADDR	CODE	STMT	SOURCE	STATEMENT
*0046	DDA605	0060	AND	(IX+IND)
*0049	FDA605	0061	AND	(IY+IND)
*004C	A7	0062	AND	A
*004D	A0	0063	AND	B
*004E	A1	0064	AND	C
*004F	A2	0065	AND	D
*0050	A3	0066	AND	E
*0051	A4	0067	AND	H
*0052	A5	0068	AND	L
*0053	E620	0069	AND	N
		0070 ;		
*0055	CB46	0071	BIT	0, (HL) ; TEST BIT 0 IN BYTE ADDR BY HL
*0057	DDCB0546	0072	BIT	0, (IX+IND)
*005B	FDCB0546	0073	BIT	0, (IY+IND)
*005F	CB47	0074	BIT	0, A
*0061	CB40	0075	BIT	0, B
*0063	CB41	0076	BIT	0, C
*0065	CB42	0077	BIT	0, D
*0067	CB43	0078	BIT	0, E
*0069	CB44	0079	BIT	0, H
*006B	CB45	0080	BIT	0, L
		0081 ;		
*006D	CB4E	0082	BIT	1, (HL)
*006F	DDCB054E	0083	BIT	1, (IX+IND)
*0073	FDCB054E	0084	BIT	1, (IY+IND)
*0077	CB4F	0085	BIT	1, A
*0079	CB48	0086	BIT	1, B
*007B	CB49	0087	BIT	1, C
*007D	CB4A	0088	BIT	1, D
*007F	CB4B	0089	BIT	1, E
*0081	CB4C	0090	BIT	1, H
*0083	CB4D	0091	BIT	1, L
		0092 ;		
*0085	CB56	0093	BIT	2, (HL)
*0087	DDCB0556	0094	BIT	2, (IX+IND)
*008B	FDCB0556	0095	BIT	2, (IY+IND)
*008F	CB57	0096	BIT	2, A
*0091	CB50	0097	BIT	2, B
*0093	CB51	0098	BIT	2, C
*0095	CB52	0099	BIT	2, D
*0097	CB53	0100	BIT	2, E
*0099	CB54	0101	BIT	2, H
*009B	CB55	0102	BIT	2, L
		0103 ;		
*009D	CB5E	0104	BIT	3, (HL)
*009F	DDCB055E	0105	BIT	3, (IX+IND)
*00A3	FDCB055E	0106	BIT	3, (IY+IND)
*00A7	CB5F	0107	BIT	3, A
*00A9	CB58	0108	BIT	3, B
*00AB	CB59	0109	BIT	3, C
*00AD	CB5A	0110	BIT	3, D
*00AF	CB5B	0111	BIT	3, E
*00B1	CB5C	0112	BIT	3, H
*00B3	CB5D	0113	BIT	3, L
		0114 ;		
*00B5	CB66	0115	BIT	4, (HL)
*00B7	DDCB0566	0116	BIT	4, (IX+IND)
*00BB	FDCB0566	0117	BIT	4, (IY+IND)

ADDR	CODE	STMT	SOURCE	STATEMENT
0005	CB67	0118	BIT	4,A
0006	CB60	0119	BIT	4,B
000C3	CB61	0120	BIT	4,C
000C5	CB62	0121	BIT	4,D
000C7	CB63	0122	BIT	4,E
000C9	CB64	0123	BIT	4,H
000CB	CB65	0124	BIT	4,L
		0125 ;		
000CD	CB6E	0126	BIT	5,(HL)
000CF	DDCB056E	0127	BIT	5,(IX+IND)
000D3	FDCB056E	0128	BIT	5,(IY+IND)
000D7	CB6F	0129	BIT	5,A
000D9	CB68	0130	BIT	5,B
000DB	CB69	0131	BIT	5,C
000DD	CB6A	0132	BIT	5,D
000DF	CB6B	0133	BIT	5,E
000E1	CB6C	0134	BIT	5,H
000E3	CB6D	0135	BIT	5,L
		0136 ;		
000E5	CB76	0137	BIT	6,(HL)
000E7	DDCB0576	0138	BIT	6,(IX+IND)
000EB	FDCB0576	0139	BIT	6,(IY+IND)
000EF	CB77	0140	BIT	6,A
000F1	CB70	0141	BIT	6,B
000F3	CB71	0142	BIT	6,C
000F5	CB72	0143	BIT	6,D
000F7	CB73	0144	BIT	6,E
000F9	CB74	0145	BIT	6,H
000FB	CB75	0146	BIT	6,L
		0147 ;		
000FD	CB7E	0148	BIT	7,(HL)
000FF	DDCB057E	0149	BIT	7,(IX+IND)
0103	FDCB057E	0150	BIT	7,(IY+IND)
0107	CB7F	0151	BIT	7,A
0109	CB78	0152	BIT	7,B
010B	CB79	0153	BIT	7,C
010D	CB7A	0154	BIT	7,D
010F	CB7B	0155	BIT	7,E
0111	CB7C	0156	BIT	7,H
0113	CB7D	0157	BIT	7,L
		0158 ;		
0115	DC0700	0159	CALL	C,NN ;CALL SUBROUTINE AT NN IF CARRY=1
0118	FC0700	0160	CALL	M,NN ;CALL IF RESULT MINUS
011B	D40700	0161	CALL	NC,NN ;CALL IF NO CARRY
011E	CD0700	0162	CALL	NN ;UNCONDITIONAL CALL
0121	C40700	0163	CALL	NZ,NN ;CALL IF RESULT NONZERO
0124	F40700	0164	CALL	P,NN ;CALL IF RESULT POSITIVE
0127	EC0700	0165	CALL	PE,NN ;IF PARITY EVEN
012A	E40700	0166	CALL	PO,NN ;IF PARITY ODD
012D	CC0700	0167	CALL	Z,NN ;IF RESULTS ZERO
		0168 ;		
0130	3F	0169	CCF	;COMPLEMENT CARRY FLAG
		0170 ;		
0131	BE	0171	CP	(HL) ;COMPARE BYTE ADDR BY HL WITH A
		0172		;IF EQUAL, Z=1 (RESULTS ZERO)
		0173		;IF A GREATER, RESULTS POSITIVE
		0174		;IF A SMALLER, RESULTS NEGATIVE.(CARRY=1)
0132	DDBE05	0175	CP	(IX+IND)

ADDR	CODE	STMT	SOURCE	STATEMENT
*0135	FD8E05	0176	CP	(IY+IND)
*0138	BF	0177	CP	A
*0139	B8	0178	CP	B
*013A	B9	0179	CP	C
*013B	BA	0180	CP	D
*013C	BB	0181	CP	E
*013D	BC	0182	CP	H
*013E	BD	0183	CP	L
*013F	FE20	0184	CP	N
		0185 ;		
*0141	EDA9	0186	CPD	;COMPARE BYTE AT (HL) WITH A, DECR HL & BC
*0143	EDB9	0187	CPDR	;AS ABOVE, REPEAT UNTIL BC=0
*0145	EDA1	0188	CPI	;AS ABOVE, INCR HL, DECR BC, NO REPEAT
*0147	EDB1	0189	CPIR	;AS IN CPI, REPEAT UNTIL BC=0
		0190 ;		
*0149	2F	0191	CPL	;1'S COMPEMENT A
		0192 ;		
*014A	27	0193	DAA	;DECIMALLY ADJUST A (MAKE A BCD NUMBER)
		0194 ;		
*014B	35	0195	DEC	(HL) ;DECREMENT BYTE AT (HL)
*014C	DD3505	0196	DEC	(IX+IND)
*014F	FD3505	0197	DEC	(IY+IND)
*0152	3D	0198	DEC	A
*0153	05	0199	DEC	B
*0154	0B	0200	DEC	BC
*0155	0D	0201	DEC	C
*0156	15	0202	DEC	D
*0157	1B	0203	DEC	DE
*0158	1D	0204	DEC	E
*0159	25	0205	DEC	H
*015A	2B	0206	DEC	HL
*015B	DD2B	0207	DEC	IX
*015D	FD2B	0208	DEC	IY
*015F	2D	0209	DEC	L
*0160	3B	0210	DEC	SP
		0211 ;		
*0161	F3	0212	DI	;DISABLE INTERRUPTS
		0213 ;		
*0162	102E	0214	DJNZ	DIS ;DECR. B, ADD 'DIS' TO PC IF B<>0
		0215 ;		
*0164	FB	0216	EI	;ENABLE INTERRUPTS
		0217 ;		
*0165	E3	0218	EX	(SP),HL ;EXCHANGE BYTE POINTED TO
		0219		;BY STACK POINTER WITH HL
*0166	DDE3	0220	EX	(SP),IX
*0168	FDE3	0221	EX	(SP),IY
*016A	0B	0222	EX	AF,AF' ;EXCHANGE A WITH ALTERNATE A REG
*016B	EB	0223	EX	DE,HL
*016C	D9	0224	EXX	;EXCHANGE PRIMARY REG SET WITH ALTERNATE
		0225 ;		
*016D	76	0226	HALT	;WAIT FOR INTERRUPT OR RESET
		0227 ;		
*016E	ED46	0228	IM	0 ;SET INTERRUPT MODE TO 0
*0170	ED56	0229	IM	1
*0172	ED5E	0230	IM	2
		0231 ;		
*0174	ED78	0232	IN	A, (C) ;INPUT TO A FROM PORT SPEC'D BY C
*0176	DB20	0233	IN	A, (N) ;AS ABOVE FROM PORT N,

ADDR	CODE	STMT	SOURCE	STATEMENT
------	------	------	--------	-----------

		0234		; WHERE N CAN TAKE ANY VALUE
0	ED40	0235	IN	B, (C)
017A	ED48	0236	IN	C, (C)
017C	ED50	0237	IN	D, (C)
017E	ED58	0238	IN	E, (C)
0180	ED70	0239	IN	F, (C)
0182	ED60	0240	IN	H, (C)
0184	ED68	0241	IN	L, (C)
		0242 ;		
0186	34	0243	INC	(HL) ; INCREMENT BYTE AT (HL)
0187	FD3405	0244	INC	(IY+IND)
018A	DD3405	0245	INC	(IX+IND)
018D	3C	0246	INC	A
018E	04	0247	INC	B
018F	03	0248	INC	BC
0190	0C	0249	INC	C
0191	14	0250	INC	D
0192	13	0251	INC	DE
0193	1C	0252	INC	E
0194	24	0253	INC	H
0195	23	0254	INC	HL
0196	DD23	0255	INC	IX
0198	FD23	0256	INC	IY
019A	2C	0257	INC	L
019B	33	0258	INC	SP
		0259 ;		
0	EDAA	0260	IND	; LOAD BYTE AT (HL) WITH INPUT
		0261		; FROM PORT (C), DECR. HL AND B
019E	EDBA	0262	INDR	; AS ABOVE, REPEAT UNTIL B=0
01A0	EDA2	0263	INI	; AS ABOVE, INCR. HL, DECR. B, NO REPEAT
01A2	EDB2	0264	INIR	; AS INI, REPEAT UNTIL B=0
		0265 ;		
01A4	E9	0266	JP	(HL) ; JUMP TO ADDRESS IN HL
01A5	DDE9	0267	JP	(IX)
01A7	FDE9	0268	JP	(IY)
01A9	DA0700'	0269	JP	C, NN
01AC	FA0700'	0270	JP	M, NN
01AF	D20700'	0271	JP	NC, NN
01B2	C30700'	0272	JP	NN
01B5	C20700'	0273	JP	NZ, NN
01B8	F20700'	0274	JP	P, NN
01BB	EA0700'	0275	JP	PE, NN
01BE	E20700'	0276	JP	PO, NN
01C1	CA0700'	0277	JP	Z, NN
		0278 ;		
01C4	382E	0279	JR	C, DIS
01C6	182E	0280	JR	DIS ; ADD 'DIS' TO PC (JUMP RELATIVE)
01C8	302E	0281	JR	NC, DIS ; DIS=2EH FOR ASSEMBLY
01CA	202E	0282	JR	NZ, DIS
01CC	282E	0283	JR	Z, DIS
		0284 ;		
0	02	0285	LD	(BC), A ; LOAD BYTE AT (BC) WITH A
01CF	12	0286	LD	(DE), A
01D0	77	0287	LD	(HL), A
01D1	70	0288	LD	(HL), B
01D2	71	0289	LD	(HL), C
01D3	72	0290	LD	(HL), D
01D4	73	0291	LD	(HL), E

ADDR	CODE	STMT	SOURCE	STATEMENT
*01D5	74	0292	LD	(HL),H
*01D6	75	0293	LD	(HL),L
*01D7	3620	0294	LD	(HL),N
		0295 ;		
*01D9	DD7705	0296	LD	(IX+IND),A ;LOAD BYTE AT (IX)
		0297		;PLUS 'IND' WITH A
*01DC	DD7005	0298	LD	(IX+IND),B
*01DF	DD7105	0299	LD	(IX+IND),C
*01E2	DD7205	0300	LD	(IX+IND),D
*01E5	DD7305	0301	LD	(IX+IND),E
*01E8	DD7405	0302	LD	(IX+IND),H
*01EB	DD7505	0303	LD	(IX+IND),L
*01EE	DD360520	0304	LD	(IX+IND),N
		0305 ;		
*01F2	FD7705	0306	LD	(IY+IND),A
*01F5	FD7005	0307	LD	(IY+IND),B
*01F8	FD7105	0308	LD	(IY+IND),C
*01FB	FD7205	0309	LD	(IY+IND),D
*01FE	FD7305	0310	LD	(IY+IND),E
*0201	FD7405	0311	LD	(IY+IND),H
*0204	FD7505	0312	LD	(IY+IND),L
*0207	FD360520	0313	LD	(IY+IND),N
		0314 ;		
*020B	320700'	0315	LD	(NN),A ;STORE A AT LOCATION NN
*020E	ED430700'	0316	LD	(NN),BC
*0212	ED530700'	0317	LD	(NN),DE
*0216	220700'	0318	LD	(NN),HL
*0219	DD220700'	0319	LD	(NN),IX
*021D	FD220700'	0320	LD	(NN),IY
*0221	ED730700'	0321	LD	(NN),SP
		0322 ;		
*0225	0A	0323	LD	A,(BC) ;LOAD A FROM BYTE ADDR BY BC
*0226	1A	0324	LD	A,(DE)
*0227	7E	0325	LD	A,(HL)
*0228	DD7E05	0326	LD	A,(IX+IND)
*022B	FD7E05	0327	LD	A,(IY+IND)
*022E	3A0700'	0328	LD	A,(NN)
*0231	7F	0329	LD	A,A
*0232	78	0330	LD	A,B
*0233	79	0331	LD	A,C
*0234	7A	0332	LD	A,D
*0235	7B	0333	LD	A,E
*0236	7C	0334	LD	A,H
*0237	ED57	0335	LD	A,I
*0239	7D	0336	LD	A,L
*023A	3E20	0337	LD	A,N
*023C	ED5F	0338	LD	A,R
		0339 ;		
*023E	46	0340	LD	B,(HL)
*023F	DD4605	0341	LD	B,(IX+IND)
*0242	FD4605	0342	LD	B,(IY+IND)
*0245	47	0343	LD	B,A
*0246	40	0344	LD	B,B
*0247	41	0345	LD	B,C
*0248	42	0346	LD	B,D
*0249	43	0347	LD	B,E
*024A	44	0348	LD	B,H
*024B	45	0349	LD	B,L

ADDR	CODE	STMT	SOURCE	STATEMENT
*024C	0620	0350	LD	B,N
		0351 ;		
*024E	ED4B0700	0352	LD	BC, (NN)
*0252	010700	0353	LD	BC, NN
		0354 ;		
*0255	4E	0355	LD	C, (HL)
*0256	DD4E05	0356	LD	C, (IX+IND)
*0259	FD4E05	0357	LD	C, (IY+IND)
*025C	4F	0358	LD	C, A
*025D	48	0359	LD	C, B
*025E	49	0360	LD	C, C
*025F	4A	0361	LD	C, D
*0260	4B	0362	LD	C, E
*0261	4C	0363	LD	C, H
*0262	4D	0364	LD	C, L
*0263	0E20	0365	LD	C, N
		0366 ;		
*0265	56	0367	LD	D, (HL)
*0266	DD5605	0368	LD	D, (IX+IND)
*0269	FD5605	0369	LD	D, (IY+IND)
*026C	57	0370	LD	D, A
*026D	50	0371	LD	D, B
*026E	51	0372	LD	D, C
*026F	52	0373	LD	D, D
*0270	53	0374	LD	D, E
*0271	54	0375	LD	D, H
*0272	55	0376	LD	D, L
*0273	1620	0377	LD	D, N
		0378 ;		
*0275	ED5B0700	0379	LD	DE, (NN)
*0279	110700	0380	LD	DE, NN
		0381 ;		
*027C	5E	0382	LD	E, (HL)
*027D	DD5E05	0383	LD	E, (IX+IND)
*0280	FD5E05	0384	LD	E, (IY+IND)
*0283	5F	0385	LD	E, A
*0284	58	0386	LD	E, B
*0285	59	0387	LD	E, C
*0286	5A	0388	LD	E, D
*0287	5B	0389	LD	E, E
*0288	5C	0390	LD	E, H
*0289	5D	0391	LD	E, L
*028A	1E20	0392	LD	E, N
		0393 ;		
*028C	66	0394	LD	H, (HL)
*028D	DD6605	0395	LD	H, (IX+IND)
*0290	FD6605	0396	LD	H, (IY+IND)
*0293	67	0397	LD	H, A
*0294	60	0398	LD	H, B
*0295	61	0399	LD	H, C
*0296	62	0400	LD	H, D
*0297	63	0401	LD	H, E
*0298	64	0402	LD	H, H
*0299	65	0403	LD	H, L
*029A	2620	0404	LD	H, N
		0405 ;		
*029C	2A0700	0406	LD	HL, (NN)
*029F	210700	0407	LD	HL, NN

ADDR	CODE	STMT	SOURCE	STATEMENT
		0408 ;		
*02A2	ED47	0409	LD	I,A
		0410 ;		
02A4	DD2A0700	0411	LD	IX,(NN)
02A8	DD210700	0412	LD	IX,NN
		0413 ;		
02AC	FD2A0700	0414	LD	IY,(NN)
02B0	FD210700	0415	LD	IY,NN
		0416 ;		
*02B4	6E	0417	LD	L,(HL)
*02B5	DD6E05	0418	LD	L,(IX+IND)
*02B8	FD6E05	0419	LD	L,(IY+IND)
*02BB	6F	0420	LD	L,A
*02BC	68	0421	LD	L,B
*02BD	69	0422	LD	L,C
*02BE	6A	0423	LD	L,D
*02BF	6B	0424	LD	L,E
*02C0	6C	0425	LD	L,H
*02C1	6D	0426	LD	L,L
*02C2	2E20	0427	LD	L,N
		0428 ;		
*02C4	ED4F	0429	LD	R,A
		0430 ;		
02C6	ED7B0700	0431	LD	SP,(NN)
*02CA	F9	0432	LD	SP,HL
*02CB	DDF9	0433	LD	SP,IX
*02CD	FDF9	0434	LD	SP,IY
02CF	310700	0435	LD	SP,NN
		0436 ;		
*02D2	EDAB	0437	LDD	;LOAD BYTE AT (DE) WITH BYTE AT (HL)
		0438		;DECR. DE, HL, BC
*02D4	EDBB	0439	LDDR	;AS ABOVE, REPEAT UNTIL BC=0.
*02D6	EDA0	0440	LDI	;AS LDD, BUT INCR. DE, HL, DECR. BC
*02D8	EDB0	0441	LDIR	;AS LDI, REPEAT UNTIL BC=0
		0442 ;		
*02DA	ED44	0443	NEG	;2'S COMPLEMENT A
		0444 ;		
*02DC	00	0445	NOF	;NO-OP (DO NOTHING)
		0446 ;		
*02DD	B6	0447	OR	(HL) ;LOGICAL 'OR' A AND BYTE AT (HL)
*02DE	DDB605	0448	OR	(IX+IND)
*02E1	FDB605	0449	OR	(IY+IND)
*02E4	B7	0450	OR	A
*02E5	B0	0451	OR	B
*02E6	B1	0452	OR	C
*02E7	B2	0453	OR	D
*02E8	B3	0454	OR	E
*02E9	B4	0455	OR	H
*02EA	B5	0456	OR	L
*02EB	F620	0457	OR	N
		0458 ;		
*02ED	EDBB	0459	OTDR	;LOAD OUTPUT PORT (C) WITH BYTE AT (HL)
		0460		;DECR. HL AND B, REPEAT UNTIL B=0
*02EF	EDB3	0461	OTIR	;AS ABOVE, BUT INCR. HL
		0462 ;		
*02F1	ED79	0463	OUT	(C),A ;OUTPUT A TO PORT SPEC'D BY C
*02F3	ED41	0464	OUT	(C),B
*02F5	ED49	0465	OUT	(C),C

ADDR	CODE	STMT	SOURCE	STATEMENT
*02F7	ED51	0466	OUT	(C),D
*02F8	ED59	0467	OUT	(C),E
*02FB	ED61	0468	OUT	(C),H
*02FD	ED69	0469	OUT	(C),L
*02FF	D320	0470	OUT	(N),A ; OUTPUT A TO PORT N
		0471 ;		
*0301	EDAB	0472	OUTD	; AS OTDR, BUT NO REPEAT
*0303	EDA3	0473	OUTI	; AS OTIR, BUT NO REPEAT
		0474 ;		
*0305	F1	0475	POP	AF ; RETRIEVE A FROM STACK
*0306	C1	0476	POP	BC
*0307	D1	0477	POP	DE
*0308	E1	0478	POP	HL
*0309	DDE1	0479	POP	IX
*030B	FDE1	0480	POP	IY
		0481 ;		
*030D	F5	0482	PUSH	AF ; PUT A ON STACK
*030E	C5	0483	PUSH	BC
*030F	D5	0484	PUSH	DE
*0310	E5	0485	PUSH	HL
*0311	DDE5	0486	PUSH	IX
*0313	FDE5	0487	PUSH	IY
		0488 ;		
*0315	CB86	0489	RES	0, (HL) ; RESET (MAKE 0) BIT 0
		0490		; OF BYTE AT (HL)
*0317	DDCB0586	0491	RES	0, (IX+IND)
*0318	FDCB0586	0492	RES	0, (IY+IND)
*031F	CB87	0493	RES	0, A
*0321	CB80	0494	RES	0, B
*0323	CB81	0495	RES	0, C
*0325	CB82	0496	RES	0, D
*0327	CB83	0497	RES	0, E
*0329	CB84	0498	RES	0, H
*032B	CB85	0499	RES	0, L
		0500 ;		
*032D	CB8E	0501	RES	1, (HL)
*032F	DDCB058E	0502	RES	1, (IX+IND)
*0333	FDCB058E	0503	RES	1, (IY+IND)
*0337	CB8F	0504	RES	1, A
*0339	CB88	0505	RES	1, B
*033B	CB89	0506	RES	1, C
*033D	CB8A	0507	RES	1, D
*033F	CB8B	0508	RES	1, E
*0341	CB8C	0509	RES	1, H
*0343	CB8D	0510	RES	1, L
		0511 ;		
*0345	CB96	0512	RES	2, (HL)
*0347	DDCB0596	0513	RES	2, (IX+IND)
*034B	FDCB0596	0514	RES	2, (IY+IND)
*034F	CB97	0515	RES	2, A
*0351	CB90	0516	RES	2, B
*0353	CB91	0517	RES	2, C
*0355	CB92	0518	RES	2, D
*0357	CB93	0519	RES	2, E
*0359	CB94	0520	RES	2, H
*035B	CB95	0521	RES	2, L
		0522 ;		
*035D	CB9E	0523	RES	3, (HL)

ADDR	CODE	STMT	SOURCE	STATEMENT
*035F	DDCB059E	0524	RES	3, (IX+IND)
*0363	FDCB059E	0525	RES	3, (IY+IND)
*0367	CB9F	0526	RES	3, A
*0369	CB98	0527	RES	3, B
*036B	CB99	0528	RES	3, C
*036D	CB9A	0529	RES	3, D
*036F	CB9B	0530	RES	3, E
*0371	CB9C	0531	RES	3, H
*0373	CB9D	0532	RES	3, L
		0533 ;		
*0375	CBA6	0534	RES	4, (HL)
*0377	DDCB05A6	0535	RES	4, (IX+IND)
*037B	FDCB05A6	0536	RES	4, (IY+IND)
*037F	CBA7	0537	RES	4, A
*0381	CBA0	0538	RES	4, B
*0383	CBA1	0539	RES	4, C
*0385	CBA2	0540	RES	4, D
*0387	CBA3	0541	RES	4, E
*0389	CBA4	0542	RES	4, H
*038B	CBA5	0543	RES	4, L
		0544 ;		
*038D	CBAE	0545	RES	5, (HL)
*038F	DDCB05AE	0546	RES	5, (IX+IND)
*0393	FDCB05AE	0547	RES	5, (IY+IND)
*0397	CBAF	0548	RES	5, A
*0399	CBAB	0549	RES	5, B
*039B	CBA9	0550	RES	5, C
*039D	CBAA	0551	RES	5, D
*039F	CBAB	0552	RES	5, E
*03A1	CBAC	0553	RES	5, H
*03A3	CBAD	0554	RES	5, L
		0555 ;		
*03A5	CBB6	0556	RES	6, (HL)
*03A7	DDCB05B6	0557	RES	6, (IX+IND)
*03AB	FDCB05B6	0558	RES	6, (IY+IND)
*03AF	CBB7	0559	RES	6, A
*03B1	CBB0	0560	RES	6, B
*03B3	CBB1	0561	RES	6, C
*03B5	CBB2	0562	RES	6, D
*03B7	CBB3	0563	RES	6, E
*03B9	CBB4	0564	RES	6, H
*03BB	CBB5	0565	RES	6, L
		0566 ;		
*03BD	CBBE	0567	RES	7, (HL)
*03BF	DDCB05BE	0568	RES	7, (IX+IND)
*03C3	FDCB05BE	0569	RES	7, (IY+IND)
*03C7	CBBF	0570	RES	7, A
*03C9	CBB8	0571	RES	7, B
*03CB	CBB9	0572	RES	7, C
*03CD	CBBA	0573	RES	7, D
*03CF	CBBB	0574	RES	7, E
*03D1	CBBC	0575	RES	7, H
*03D3	CBBD	0576	RES	7, L
		0577 ;		
*03D5	C9	0578	RET	; RETURN FROM SUBROUTINE
*03D6	D8	0579	RET	C
*03D7	F8	0580	RET	M
*03D8	D0	0581	RET	NC

ADDR	CODE	STMT	SOURCE	STATEMENT
'03D9	C0	0582	RET	NZ
'0	F0	0583	RET	P
'03DB	EB	0584	RET	PE
'03DC	E0	0585	RET	PO
'03DD	CB	0586	RET	Z
		0587 ;		
'03DE	ED4D	0588	RETI	;RETURN FROM INTERRUPT ROUTINE
'03EO	ED45	0589	RETN	;RETURN FROM NONMASKABLE INTERRUPT
		0590 ;		
'03E2	CB16	0591	RL	(HL) ;ROTATE LEFT THRU CY, BYTE AT (HL)
'03E4	DDCB0516	0592	RL	(IX+IND)
'03E8	FDCB0516	0593	RL	(IY+IND)
'03EC	CB17	0594	RL	A ;ROTATE A LEFT THRU CARRY
'03EE	CB10	0595	RL	B
'03F0	CB11	0596	RL	C
'03F2	CB12	0597	RL	D
'03F4	CB13	0598	RL	E
'03F6	CB14	0599	RL	H
'03F8	CB15	0600	RL	L
		0601 ;		
'03FA	17	0602	RLA	;SAME AS RL A
		0603 ;		
'03FB	CB06	0604	RLC	(HL)
'03FD	DDCB0506	0605	RLC	(IX+IND)
'0401	FDCB0506	0606	RLC	(IY+IND)
'0405	CB07	0607	RLC	A ;ROTATE A CIRCULAR WITHOUT CY
'0	CB00	0608	RLC	B
'0409	CB01	0609	RLC	C
'040B	CB02	0610	RLC	D
'040D	CB03	0611	RLC	E
'040F	CB04	0612	RLC	H
'0411	CB05	0613	RLC	L
		0614 ;		
'0413	07	0615	RLCA	;SAME AS RLC A
		0616 ;		
'0414	ED6F	0617	RLD	;ROTATE DIGIT (4 BITS) LEFT AND RIGHT
		0618		;BETWEEN A AND (HL)
		0619 ;		
'0416	CB1E	0620	RR	(HL) ;ROTATE RT (THRU CY) BYTE AT (HL)
'0418	DDCB051E	0621	RR	(IX+IND)
'041C	FDCB051E	0622	RR	(IY+IND)
'0420	CB1F	0623	RR	A
'0422	CB18	0624	RR	B
'0424	CB19	0625	RR	C
'0426	CB1A	0626	RR	D
'0428	CB1B	0627	RR	E
'042A	CB1C	0628	RR	H
'042C	CB1D	0629	RR	L
		0630 ;		
'042E	1F	0631	RRA	;SAME AS RR A
		0632 ;		
'0	CB0E	0633	RRC	(HL) ;ROTATE RT CIRCULAR BYTE AT (HL)
'0431	DDCB050E	0634	RRC	(IX+IND)
'0435	FDCB050E	0635	RRC	(IY+IND)
'0439	CB0F	0636	RRC	A
'043B	CB08	0637	RRC	B
'043D	CB09	0638	RRC	C
'043F	CB0A	0639	RRC	D

ADDR	CODE	STMT	SOURCE	STATEMENT
*0441	CB0B	0640	RRC	E
*0443	CB0C	0641	RRC	H
*0445	CB0D	0642	RRC	L
		0643 ;		
*0447	OF	0644	RRCA	; SAME AS RRC A
		0645 ;		
*0448	ED67	0646	RRD	; ROTATE DIGIT RT AND LFT BETWEEN
		0647		; A AND LOCATION (HL)
		0648 ;		
*044A	C7	0649	RST	0 ; RESTART TO LOCATION 00
*044B	CF	0650	RST	0BH
*044C	D7	0651	RST	10H
*044D	DF	0652	RST	18H
*044E	E7	0653	RST	20H
*044F	EF	0654	RST	28H
*0450	F7	0655	RST	30H
*0451	FF	0656	RST	38H
		0657 ;		
*0452	9E	0658	SBC	A, (HL) ; SUBTRACT WITH CY THE BYTE AT
		0659		; (HL) FROM A
*0453	DD9E05	0660	SBC	A, (IX+IND)
*0456	FD9E05	0661	SBC	A, (IY+IND)
*0459	9F	0662	SBC	A, A
*045A	98	0663	SBC	A, B
*045B	99	0664	SBC	A, C
*045C	9A	0665	SBC	A, D
*045D	9B	0666	SBC	A, E
*045E	9C	0667	SBC	A, H
*045F	9D	0668	SBC	A, L
*0460	DE20	0669	SBC	A, N
		0670 ;		
*0462	ED42	0671	SBC	HL, BC ; 16 BIT SUBTRACT W/CY, BC FROM HL
*0464	ED52	0672	SBC	HL, DE
*0466	ED62	0673	SBC	HL, HL
*0468	ED72	0674	SBC	HL, SP
		0675 ;		
*046A	37	0676	SCF	; SET CARRY FLAG (CY=1)
		0677 ;		
*046B	CBC6	0678	SET	0, (HL) ; SET (TO 1) BIT 0 IN BYTE AT (HL)
*046D	DDCB05C6	0679	SET	0, (IX+IND)
*0471	FDCB05C6	0680	SET	0, (IY+IND)
*0475	CBC7	0681	SET	0, A
*0477	CBC0	0682	SET	0, B
*0479	CBC1	0683	SET	0, C
*047B	CBC2	0684	SET	0, D
*047D	CBC3	0685	SET	0, E
*047F	CBC4	0686	SET	0, H
*0481	CBC5	0687	SET	0, L
		0688 ;		
*0483	CBCE	0689	SET	1, (HL)
*0485	DDCB05CE	0690	SET	1, (IX+IND)
*0489	FDCB05CE	0691	SET	1, (IY+IND)
*048D	CBCF	0692	SET	1, A
*048F	CBC8	0693	SET	1, B
*0491	CBC9	0694	SET	1, C
*0493	CBCA	0695	SET	1, D
*0495	CBCB	0696	SET	1, E
*0497	CBCC	0697	SET	1, H

ADDR	CODE	STMT	SOURCE	STATEMENT
*0499	CBCD	0698		SET 1,L
		0699 ;		
*049B	CBD6	0700		SET 2,(HL)
*049D	DDCB05D6	0701		SET 2,(IX+IND)
*04A1	FDCB05D6	0702		SET 2,(IY+IND)
*04A5	CBD7	0703		SET 2,A
*04A7	CBD0	0704		SET 2,B
*04A9	CBD1	0705		SET 2,C
*04AB	CBD2	0706		SET 2,D
*04AD	CBD3	0707		SET 2,E
*04AF	CBD4	0708		SET 2,H
*04B1	CBD5	0709		SET 2,L
		0710 ;		
*04B3	CBDE	0711		SET 3,(HL)
*04B5	DDCB05DE	0712		SET 3,(IX+IND)
*04B9	FDCB05DE	0713		SET 3,(IY+IND)
*04BD	CBDF	0714		SET 3,A
*04BF	CBD8	0715		SET 3,B
*04C1	CBD9	0716		SET 3,C
*04C3	CBDA	0717		SET 3,D
*04C5	CBDB	0718		SET 3,E
*04C7	CBDC	0719		SET 3,H
*04C9	CBDD	0720		SET 3,L
		0721 ;		
*04CB	CBE6	0722		SET 4,(HL)
*04CD	DDCB05E6	0723		SET 4,(IX+IND)
*04D1	FDCB05E6	0724		SET 4,(IY+IND)
*04D3	CBE7	0725		SET 4,A
*04D7	CBE0	0726		SET 4,B
*04D9	CBE1	0727		SET 4,C
*04DB	CBE2	0728		SET 4,D
*04DD	CBE3	0729		SET 4,E
*04DF	CBE4	0730		SET 4,H
*04E1	CBE5	0731		SET 4,L
		0732 ;		
*04E3	CBEE	0733		SET 5,(HL)
*04E5	DDCB05EE	0734		SET 5,(IX+IND)
*04E9	FDCB05EE	0735		SET 5,(IY+IND)
*04ED	CBEF	0736		SET 5,A
*04EF	CBE8	0737		SET 5,B
*04F1	CBE9	0738		SET 5,C
*04F3	CBEA	0739		SET 5,D
*04F5	CBEB	0740		SET 5,E
*04F7	CBEC	0741		SET 5,H
*04F9	CBED	0742		SET 5,L
		0743 ;		
*04FB	CBF6	0744		SET 6,(HL)
*04FD	DDCB05F6	0745		SET 6,(IX+IND)
*0501	FDCB05F6	0746		SET 6,(IY+IND)
*0505	CBF7	0747		SET 6,A
*0507	CBF0	0748		SET 6,B
*0509	CBF1	0749		SET 6,C
*050B	CBF2	0750		SET 6,D
*050D	CBF3	0751		SET 6,E
*050F	CBF4	0752		SET 6,H
*0511	CBF5	0753		SET 6,L
		0754 ;		
*0513	CBFE	0755		SET 7,(HL)

Z80 OPCODE LISTING			SD SYSTEMS Z80 ASSEMBLER PAGE 0014	
ADDR	CODE	STMT	SOURCE	STATEMENT
0515	DDCB05FE	0756	SET	7, (IX+IND)
0519	FDCB05FE	0757	SET	7, (IY+IND)
051D	CBFF	0758	SET	7, A
051F	CBF8	0759	SET	7, B
0521	CBF9	0760	SET	7, C
0523	CBFA	0761	SET	7, D
0525	CBFB	0762	SET	7, E
0527	CBFC	0763	SET	7, H
0529	CBFD	0764	SET	7, L
		0765 ;		
052B	CB26	0766	SLA	(HL) ;SHIFT LEFT ARITHMETIC,
		0767		;BYTE AT (HL)
052D	DDCB0526	0768	SLA	(IX+IND)
0531	FDCB0526	0769	SLA	(IY+IND)
0535	CB27	0770	SLA	A
0537	CB20	0771	SLA	B
0539	CB21	0772	SLA	C
053B	CB22	0773	SLA	D
053D	CB23	0774	SLA	E
053F	CB24	0775	SLA	H
0541	CB25	0776	SLA	L
		0777 ;		
0543	CB2E	0778	SRA	(HL) ;SHIFT RT ARITH., BYTE AT (HL)
0545	DDCB052E	0779	SRA	(IX+IND)
0549	FDCB052E	0780	SRA	(IY+IND)
054D	CB2F	0781	SRA	A
054F	CB28	0782	SRA	B
0551	CB29	0783	SRA	C
0553	CB2A	0784	SRA	D
0555	CB2B	0785	SRA	E
0557	CB2C	0786	SRA	H
0559	CB2D	0787	SRA	L
		0788 ;		
055B	CB3E	0789	SRL	(HL) ;SHIFT RT LOGICAL, BYTE AT (HL)
055D	DDCB053E	0790	SRL	(IX+IND)
0561	FDCB053E	0791	SRL	(IY+IND)
0565	CB3F	0792	SRL	A
0567	CB38	0793	SRL	B
0569	CB39	0794	SRL	C
056B	CB3A	0795	SRL	D
056D	CB3B	0796	SRL	E
056F	CB3C	0797	SRL	H
0571	CB3D	0798	SRL	L
		0799 ;		
0573	96	0800	SUB	(HL) ;SUBTRACT (NO CARRY) BYTE AT
		0801		; (HL) FROM A
0574	DD9605	0802	SUB	(IX+IND)
0577	FD9605	0803	SUB	(IY+IND)
057A	97	0804	SUB	A
057B	90	0805	SUB	B
057C	91	0806	SUB	C
057D	92	0807	SUB	D
057E	93	0808	SUB	E
057F	94	0809	SUB	H
0580	95	0810	SUB	L
0581	D620	0811	SUB	N
		0812 ;		
0583	AE	0813	XOR	(HL) ;"EXCLUSIVE OR" BYTE AT

ADDR CODE STMT SOURCE STATEMENT

```
001 DDAE05 0814 ;(HL) WITH A
0587 FDAE05 0815 XOR (IX+IND)
058A AF 0816 XOR (IY+IND)
058B AB 0817 XOR A
058C A9 0818 XOR B
058D AA 0819 XOR C
058E AB 0820 XOR D
058F AC 0821 XOR E
0590 AD 0822 XOR H
0591 EE20 0823 XOR L
0824 ;EXCLUSIVE OR A WITH VALUE 'N'
0825 ;
0826 END
```

ERRORS=0000

THE BIT FIDDLERS
SOFTWARE PROBLEM REPORT

Please use this form to report errors or problems in software supplied by The Bit Fiddlers. This form is designed to act as a transmittal sheet, and problem details can be described on additional pages.

Date:.....

Software Product Name:..... Version No.:.....

Computer Type:..... Memory Size:.....

Operating System:..... Version No.:.....

Number of Disk Drives:.....

Please describe the problem you have encountered. Include references to the manual if appropriate. Try to reduce the problem to a simple test case. Enclose any appropriate listings. If you have discovered a patch or interim solution please describe it.

This form may also be used to suggest enhancements to our software products.

PROBLEM DESCRIPTION:

Name:..... Phone:.....

Address:.....

City:..... State:..... Zip:.....

Return to: THE BIT FIDDLERS
P.O. Box 11023
San Diego, CA 92111-0010

